ＦＣＡＤ５１６ＰＣＩ－１６ＭＷ
取扱説明書

１６チャンネル逐次サンプリング

１６ＭＷバッファメモリ内蔵

１６ビット高速アナログ変換基板

対応パソコン
IBM PC/AT互換機 （PCI）

２０１５年１０月２１日　マイクロサイエンス互換ドライバーパッケージ説明追加
２０１５年7月９日　アナログレンジ説明修正
２０１５年3月３日　ドライバーアンインストール操作説明追加
２０１５年1月２３日発行　初版

フレックスコア

合同会社　フレックスコア

目次

本製品の使用・適用についての注意事項 5
故障・修理・サポート方法について 6
本製品の構成 6
第１章　導入・試運転 8
１－１　本製品の概要 8
１－２　ボード上の設定 9
１－３　入出力コネクタ・ピン接続 10
アナログ入力　プラグ：１７ＪＥ－２３３７０－０２（Ｄ８Ａ）　ＤＤＫ製・添付 10
デジタルコネクタ 11
１－４　ボードのインストール 11
１－５　動作確認・試運転 11
第２章　信号入出力 12
２－１　アナログ入力回路 12
２－２　アナログ入力範囲 12
２－３　アナログ入力特性（誤差、ドリフト、雑音、保護対策） 13
２－４　外部サンプルホールド制御 14
２－５　デジタル入出力回路 14
第３章　制御・操作 15
３－１　ＡＤサンプリング動作、トリガ動作の様子 15
有限自動サンプリングモード 15
無限自動サンプリングモード 15
マニュアルサンプリングモード 15
３－２　オンボードメモリとユーザーアプリケーションとの関係性 15
３－３　制御レジスタアドレスマップ 17
３－４　ボード制御部リセット 18
３－５　アナログ回路接続設定・読出し 18
３－６　データコード指定 20
３－７　アナログ入力スキャン最終チャンネル指定 20
３－８　クロック源選択 21
３－９　クロック源分周比設定 21
３－10　アナログトリガレベル設定 21
３－11　トリガモード設定（ソフトトリガ実行を含む） 22
３－12　マニュアルサンプリングスキャン 22
３－13　ステータス入力 23
３－14　ステータスクリア操作 24
３－15　ＦＩＦＯメモリ保持データ数読込み 24
３－16　ＡＤデータ読出し 24
３－17　割り込み制御（要因設定） 25
３－18　割り込み要求クリア 25
３－19　マスタースレーブ接続 26
ボード上の設定 26
ボード間の接続 26
ソフトウェア設定 26
マルチサンプリングとの関係 26
３－20　汎用デジタル入力 26
３－21　汎用デジタル出力 27

３－22　外付け同時サンプルホールド制御 27
３－23　マルチサンプリング回数指定 27
３－24　マルチサンプリングクロック源の選択 28
３－25　マルチサンプリングクロック源分周比指定 28
第４章　ソフトウェア 29
４－１　本製品用ソフトウェアの種類 29
４－２　ドライバーのインストール 30
４－２－１　WindowsXP ドライバーインストール及びアンインストール 30
４－２－２　Windows 7 （X86：32 ビット）へのインストール 33
４－２－３　Windows 7 （X64：64 ビット）へのインストール 36
４－３　Windows 7 ドライバーのアンインストール 39
４－４　サンプルプログラムについて 41
４－５　サンプルプログラム、動作確認プログラムについて（補足） 41
４－６　関数仕様 43
【１】ボード及び本ハンドラーの初期化 44
【２】サンプリングチャンネル数設定 44
【３】サンプリングモード指定 44
【４】トリガ関連の設定 44
【５】レンジトリガ設定 45
【６】マルチサンプリング動作モード設定 45
【７】サンプリングクロックオプション或いは外部クロック源周波数の設定 45
【８】クロック源及びクロック値の設定 45
【９】マルチサンプリング用外部クロック値の設定 46
【10】マルチサンプリングクロック値の設定 46
【11】サンプリング開始 46
【12】ハンドラー終了 46
【13】ステータス取得 47
【14】Ｄｌｌバッファーからのデータ転送 47
【15】ボードからユーザープログラム内バッファーへの直接転送 47
【16】マニュアルサンプリングスキャン 47
【17】汎用デジタル出力 48
【18】汎用デジタル入力 48
【19】Ｄｌｌバッファーのリングバッファ化制御 48
【20】アナログ入力モードの設定 48
【21】サンプリング動作の強制停止 48
【22】各フラグのクリア 49
【23】残りＡＤデータの読込み 49
【24】外付・同時サンプルホールド制御 49
【25】サンプリングスキャン速度指定 49
【26】オプションメッセージ発信の設定 49
【27】本ハンドラーのバージョン取得 50
FCAD516 独自機能用関数セット 50
【28】ハーフフルサイズの取得 50
【29】使用済みメモリ量の取得 50
【30】アナログ回路条件の設定 50
【31】ADデータコード、入力モード等の取得 51
４－７　エラーコード 52

第５章　汎用デバイスドライバー仕様書 53
【１】ドライバーオープン 54
【２】ドライバースタート 54
【３】バイトリード 54
【４】バイトブロックリード 54
【５】ワードリード 55
【６】ワードブロックリード 55
【７】ダブルワードリード 55
【８】ダブルワードブロックリード 55
【９】バイトライト 56
【10】バイトブロックライト 56
【11】ワードライト 56
【12】ワードブロックライト 56
【13】ダブルワードライト 57
【14】ダブルワードブロックライト 57
【15】ドライバーストップ 57
【16】ドライバークローズ 57
エラーコード 57
第６章　保守、その他 58
６－１　故障・トラブル等の原因と対処 58
再点検、確認ポイント 58
ボードID 58
トリガ方法 58
デジタル入出力 58
アナログ入力 58
６－２　修理のときは 58
６－３　アナログ入力範囲の再調整 59
６－４　外部制御信号、タイミング等 59

本製品の使用・適用についての注意事項

（1）　本製品は、ＩＢＭＰＣ／ＡＴ互換機のＰＣＩバス拡張Ｉ／Ｏスロット、またはＰＣＩバス拡張Ｉ／Ｏボックスに装着して使用するものです。

（2）　本製品が組み込まれたシステムの運用対象・方法・場所・環境等によって、故障・誤動作等が生じた場合に起こり得る、身体・生命・財産等に対する損害の回避処置は同システムの設計・製作に別途付加・反映させて下さい。本製品自体には、前述の機能はなく、従って弊社では本製品が組み込まれたシステムの運用により発生した故障・誤動作・事故に起因する身体・生命・財産等の損害に対する責任は負えません。これは本製品の故障・誤動作が原因となった場合も含み、理由の如何を問いません。

（3）　本製品付属のソフトウェアは本製品利用の方法を示す例、またオプションの関連ソフトウェアは本製品利用の一般的便宜をはかるものであり、現在未発見のバグ存在の可能性も含めて、運用結果についての責任は一切負えません。
これらのソフトウェアには自身が組み込まれたシステムに故障・誤動作・事故等が生じた場合に起こり得る身体・生命・財産等に対する損害の回避機能はありません。御利用の場合は同システムの設計・製作で配慮・付加・反映させて下さい。

（4）　本製品（付属ソフトウェアを含む）、及びオプションの関連ソフトウェアは医用・航空機器用・その他高信頼性・高安全性を必要とするシステムに使用しないで下さい。

（5）　本製品付属のソフトウェアについて弊社は著作権を保持しますが、第３者の権利を侵害しない限りにおいて購入者は自身が製作するシステム等に自由に組み込み、販売する事もできます。但し、弊社製ソフトウェアのソースコードを含むソフトウェアを第３者に販売･移転するときは弊社の文書による事前許可を必要とします。

（6）　弊社では、本製品の販売・サポート・保証の範囲を日本国内に限っています。

（7）　本製品は改良のため仕様変更、価格改定を行うことがあります。

故障・修理・サポート方法について

（1）　納入後１年間は自然故障､及び弊社製造上の問題に起因した事が明らかな故障製品に対して無償修理を行います。

（2）　保証期間中であっても、次の場合は有償修理となります。

ア　取扱上の不注意、誤用による故障および損傷
イ　弊社以外での修理、改造、分解掃除等による故障および損傷
ウ　泥、砂、水などのかぶり、落下、衝撃等が原因で発生した故障および損傷
エ　火災、地震、水害、落雷その他の天変地異、公害や異常電圧による故障および損傷
オ　保管上の不備（高温多湿の場所等）や手入れの不備による故障
カ　接続している他の機器に起因して生じた故障
キ　その他使用者側の責に帰する故障

（3）　修理は宅配便によるセンドバックで行います。尚運賃は互いに発送する側が負担するものとします。
出張修理は行っておりません。簡単な故障であれば一週間程度で修理・返却が可能ですが、故障状況によっては更に日数を要します。

（4）　本製品使用上の質問・トラブル対応・故障修理などは入手経路の如何にかかわらず、弊社宛に直接御相談下さい。その際は、客観情報の整理・評価を行うために必ずレポートを御送付下さい。特にEmailで情報を頂く事で解決速度が大幅に向上する事が期待できます。（support@flexcore.jp）また、内容が複雑になっている場合はfax使用も有効です。

（5）　有償修理の場合の修理費用は、基本料金￥１０，０００円＋部品代となります。簡単な故障では凡そ２万円以下程度だと思われます。修理費用限度額がある場合は、お申し付け頂ければ超過する場合に御連絡致します。

（6）　修理品送付先

〒301-0853
茨城県龍ヶ崎市松ヶ丘３丁目１８番地３
フレックスコア
品質管理部　林
E-mail: support@flexcore.jp
Fax:050-3488-3354

本製品の構成

本製品は　ＦＣＡＤ５１６ＰＣＩ－１６MWボード
入出力プラグ（アナログ、デジタル各１個）
からなります。
ドライバーソフトウェア、取扱説明書（本文書）、サンプルプログラムなどについては、
弊社ホームページよりダウンロードをお願いします。

本機の仕様一覧

アナログ入力部

項目	ＦＣＡＤ５１６ＰＣＩ－１６MW
入力数・信号モード	１６チャンネルシングルエンド／８チャンネル差動入力
入力範囲 （半導体マルチプレクサ制御）	±１０Ｖ／±５Ｖ／±２．５Ｖ／０～＋１０Ｖ／０～＋５Ｖ 本機の入力範囲設定回路は高精度部品を使用しているため入力範囲を切り替えても通常の用途では再調整不要の誤差範囲に収まります。
入力インピーダンス	チャンネル毎に１０MΩの終端抵抗を標準実装
クロストーク	－７８ｄＢ

ＡＤ変換部

項目		ＦＣＡＤ５１６ＰＣＩ－１６MW
分解能		１６ビット
単チャンネル サンプル速度	高速モード	５ｕＳ（２００KHｚ）
	低速モード	１０ｕＳ（１００KHｚ）
複数チャンネル サンプル速度	高速モード	５ｕＳ／Ｃｈ
	低速モード	１０ｕＳ／Ｃｈ
非直線性max		±０．００４%ＦＳ
正確度１	高速モード	±０．０２３%ＦＳ
	低速モード	
正確度２	高速モード	±０．０３５%ＦＳ
	低速モード	
内部雑音typ		±４ＬＳＢ
温度ドリフトtyp		±１０ｐｐｍ／℃
ＡＤデータコード		バイナリまたは２の補数（ソフトウェア指定）

制御部等

項目	ＦＣＡＤ５１６ＰＣＩ－１６MW
クロック	クロック源：内部１０MHｚ／内部８．１９２MHｚ／外部ＴＴＬ入力 分周機能：３２ビットプログラマブルカウンターによる分周（バイナリ）
トリガ	内部トリガ：プログラム上からの即トリガ アナログ入力（先頭チャンネル）の指定エッジ、レベルまたはレンジ 外部トリガ：外部ＴＴＬ入力の指定エッジまたはレベル
バッファメモリ	１６M語メモリ（ＦＩＦＯモード）
ＡＤデータ転送	ブロック転送（データ長はプログラムにより取得可能）若しくはワード単位転送
マスタースレーブ動作	マスターのクロック出力をスレーブに接続する事により可能
割り込み	割り込み要因：１回サンプリングスキャン終了、トリガ発生 サンプリングクロック、外部ＴＴＬ入力の指定エッジ ＦＩＦＯメモリのハーフフル遷移
汎用デジタル入出力	１ビットＴＴＬ入力、１ビットＴＴＬ出力（またはオープンコレクタ出力）
制御アドレス	組み込み対象システムのプラグアンドプレイ機能により連続アドレスを占有
基板寸法	ＰＣＩショートサイズ（１７４．３mm×９８．４mm）突起部を含まず
動作環境	周囲温度０℃～４０℃（結露しない事）、保存温度―１０℃～+８０℃（結露しない事）
付属品	入出力プラグ
電流消費	＋５Ｖ，０．９Ａ

第１章　導入・試運転

１－１　本製品の概要

本ボードは、SDRAM を使用した大容量 FIFO メモリ搭載アナログサンプリング基板で、指定された条件（クロック、トリガ、チャンネル数）に従って、自動サンプリングを行う構造となっています。ソフトウェア上では、条件設定、サンプリングスタート（トリガ待ちを含む）ステータス評価／使用メモリ量確認／データ取り込みだけで、動作タイミングなどの制御が不要です。そのため、リアルタイム制御或いは Windows 環境でのデータ処理等に最適です。

- 高精度部品の使用によりアナログ入力範囲切り替え時の再調整不要
- サンプリング点数を制限しない大容量 FIFO メモリ（１６M 語標準）
- 複数クロック源（内部１０MHｚ／内部８．１９２MHｚ／外部 TTL 入力）
- 複数トリガモード（ソフトウェア即トリガ／アナログトリガ／外部トリガ
- クロック入出力機能により複数ボードの同期運転可能

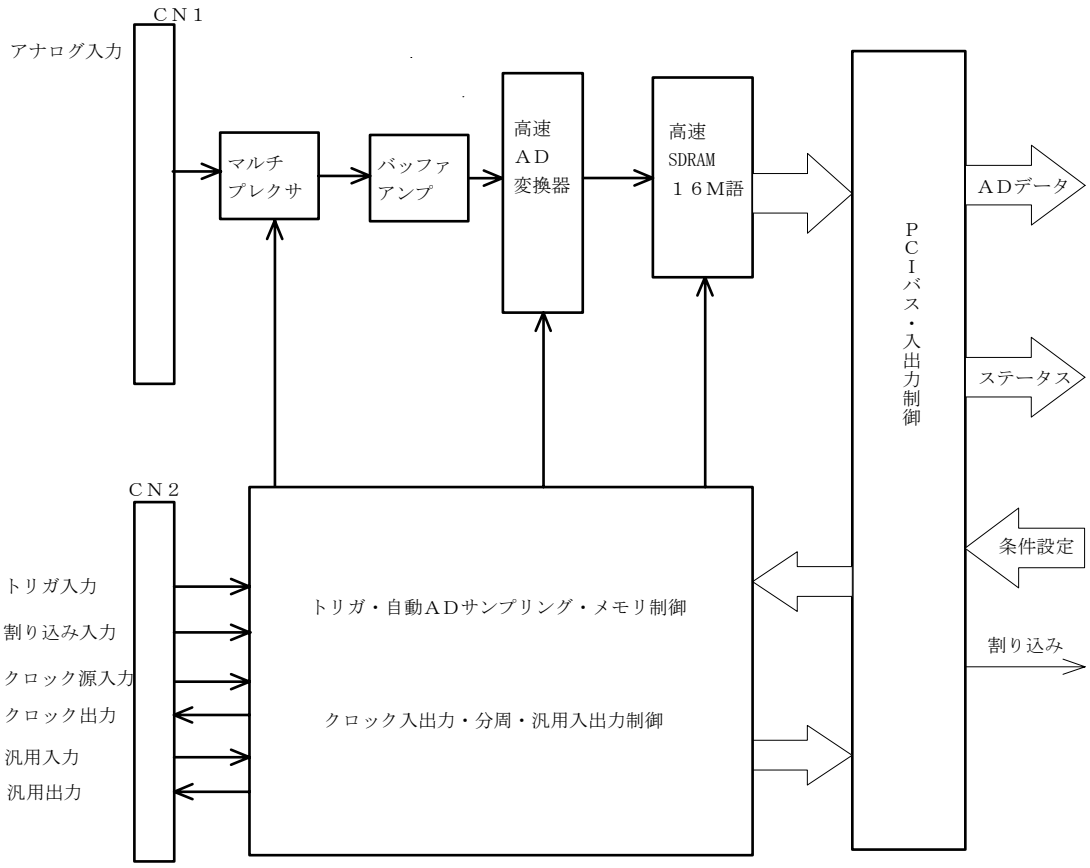


図１－１Ａ　ＦＣＡＤ５１６ＰＣＩ機能ブロック

１－２　ボード上の設定

図１－２に本ボードの外観を示します。

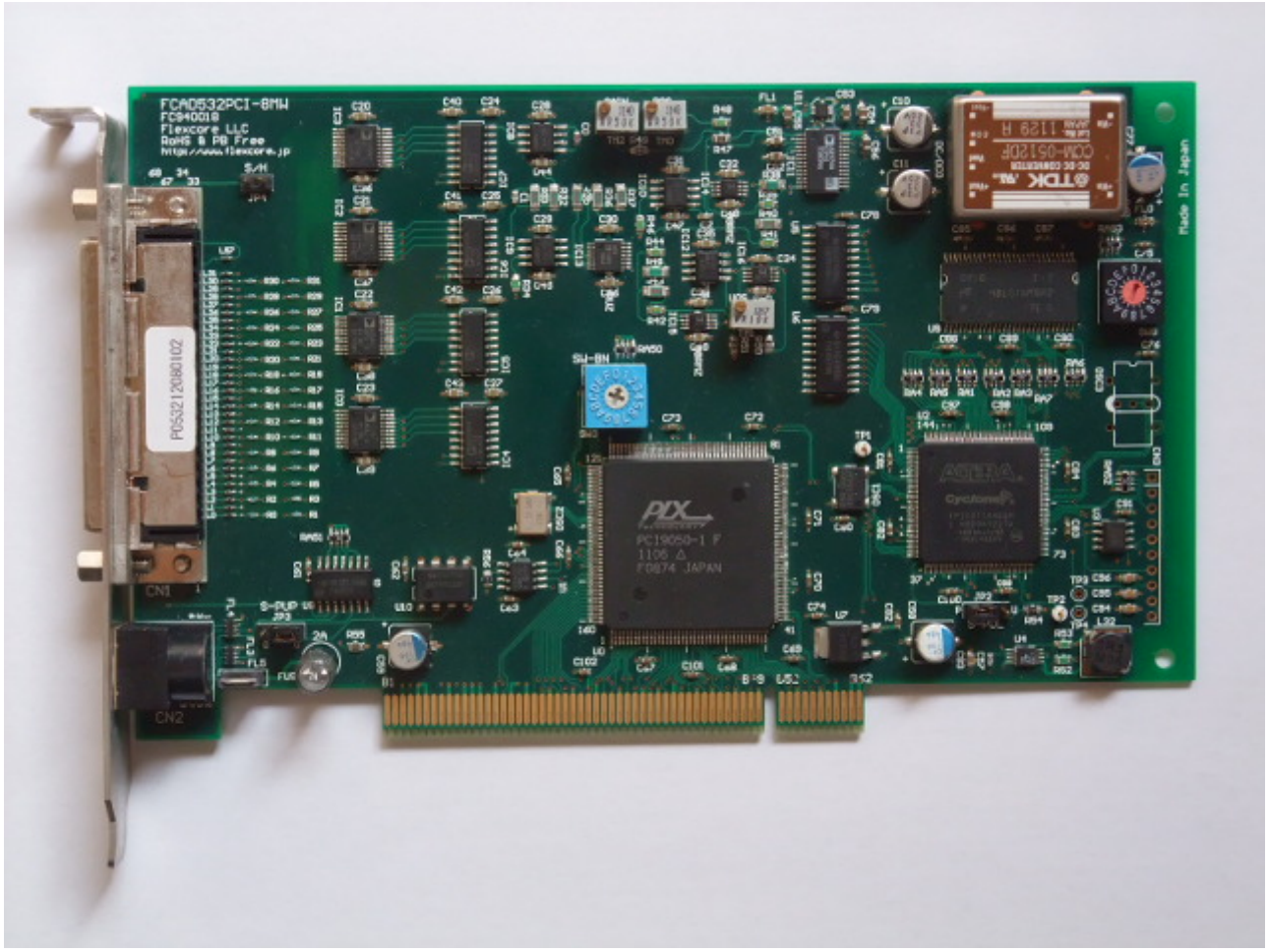


図１－２　ボード外観図

SW-BN:ボード番号設定スイッチ（出荷時：０／複数ボード使用時のボード認識用）
アナログ入力スケール選択：MUX による電子回路切り替え機能（±１０V、±５V、±２．５V等）
アナログ入力範囲極性選択：MUX による電子回路切り替え機能（バイポーラ／ユニポーラ切り替え）
アナログ入力スパン切り替え選択：MUX による電子回路切り替え機能（Aモード／B モード）
S-POL:ジャンパープラグによる機能切り替え（デジタル出力の論理：正論理／負論理）
S-PUP:ジャンパープラグによる機能切り替え（デジタル出力の構造：プルアップ抵抗有無切り替え）
S-CKZ:外部クロック入力終端
TM0:オフセット調整トリマ
TM1:ゲイン調整トリマ
TM2:バッファオフセット調整トリマ

FS1:＋５V電源出力保護ヒューズ（FRPU-2.0A：浜井電球製）
CN1:アナログ入力コネクタ（３７ピン D-SUB）
CN2:デジタル入出力コネクタ（８ピン丸型コネクタ）

１－３　入出力コネクタ・ピン接続

アナログ入力には３７ピン D-SUB コネクタ、また汎用デジタル入出力を含む制御信号入出力には８ピン丸型コネクタを使用しており適合プラグ各１個が添付されています。

アナログ入力　プラグ：１７ＪＥ－２３３７０－０２（Ｄ８Ａ）　ＤＤＫ製・添付

基板側：１７ＬＥ－１３３７０－２７（Ｄ４ＡＢ）　ＤＤＫ製・実装

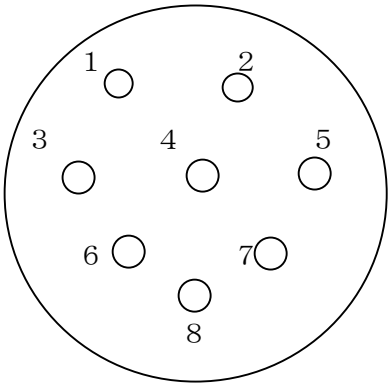
図１－３Ａ　アナログ入力コネクタ（ＣＮ１）ピン接続

信号名	機能	ピン番号	ピン番号	信号名（機能）
ＣＨ０	ＣＨ０入力	１	２０	ＡＧ（アナロググランド）
ＣＨ１	ＣＨ１入力	２	２１	ＡＧ（アナロググランド）
ＣＨ２	ＣＨ２入力	３	２２	ＡＧ（アナロググランド）
ＣＨ３	ＣＨ３入力	４	２３	ＡＧ（アナロググランド）
ＣＨ４	ＣＨ４入力	５	２４	ＡＧ（アナロググランド）
ＣＨ５	ＣＨ５入力	６	２５	ＡＧ（アナロググランド）
ＣＨ６	ＣＨ６入力	７	２６	ＡＧ（アナロググランド）
ＣＨ７	ＣＨ７入力	８	２７	ＡＧ（アナロググランド）
ＣＨ８	ＣＨ８入力	９	２８	ＡＧ（アナロググランド）
ＣＨ９	ＣＨ９入力	１０	２９	ＡＧ（アナロググランド）
ＣＨ１０	ＣＨ１０入力	１１	３０	ＡＧ（アナロググランド）
ＣＨ１１	ＣＨ１１入力	１２	３１	ＡＧ（アナロググランド）
ＣＨ１２	ＣＨ１２入力	１３	３２	ＡＧ（アナロググランド）
ＣＨ１３	ＣＨ１３入力	１４	３３	ＡＧ（アナロググランド）
ＣＨ１４	ＣＨ１４入力	１５	３４	ＡＧ（アナロググランド）
ＣＨ１５	ＣＨ１５入力	１６	３５	ＡＧ（アナロググランド）
	空きピン	１７	３６	空きピン
Ｓ／Ｈ出力	サンプルホールド制御出力	１８	３７	ＤＧ（デジタルグランド）
ＶＣＣ	＋５Ｖ出力	１９		

注　アナロググランドとデジタルグランドはボード内部で接続されています。

デジタルコネクタ

図１－３Ｃ　デジタル入出力コネクタ・ピン接続



添付プラグの半田付け側面視

１ピン：割り込み入力
２ピン：トリガ入力
３ピン：クロック入力
４ピン：クロック出力
５ピン：汎用デジタル入力
６ピン：汎用デジタル出力
７ピン：＋５Ｖ出力
８ピン：デジタルグランド

１－４　ボードのインストール

本製品はプラグアンドプレイに対応した PCI ボードです。そのため御使用に先立ち組み込むパソコンにインストール（認識、リソース割当て）する必要があります。そのためのステップは次の通りです。

１　まず、弊社ホームページよりドライバーパッケージをダウンロードします。そして、このパッケージを展開解凍し、その後インストール作業を行います。この際、本ボードはまだパソコンに取り付けてはいけません。

２　デバイスドライバーをインストールした後、一旦パソコンの電源を落とし、その後本ボードを PCI スロットに取り付けてからパソコンの電源を再投入します。一旦システムトレイにドライバーインストール中のメッセージが表示され、一分ほど待っていると、ドライバーがインストールされ、ハードウェアの使用準備ができた旨、表示されます。

３　この段階で本ボードを使用する事ができる状態になっていますので、必要に応じ適宜弊社ホームページよりサンプルプログラムをダウンロードして頂き実行モジュールを作成するか、或いはご自身で使用される言語システムを利用した開発を開始する事ができます。より詳細な内容については４－１章を参照して下さい。

１－５　動作確認・試運転

Windows7 対応の動作確認プログラム並びにサンプルプログラム（Ｃ＋）を用意しています。詳細は弊社ホームページよりダウンロードできるソフトウェアキットを参照してください。

第２章　信号入出力

２－１　アナログ入力回路

アナログ入力は、マルチプレクサで順次切り替えＡＤ変換される構造になっています。

アナログ入力範囲はボード上のＭＵＸ（ＭＵＸ１～ＭＵＸ３）によって選択され、絶対最大定格は±４０Ｖとなっています。

２－２　アナログ入力範囲

アナログ入力範囲は、ボード内ＭＵＸによって表２－２Ａの中から選択します。１６ビット分解能は、１／６５５３６ですが、公称入力範囲で正直に調整するとＡＤ変換値１デジット当たりの電圧値が割り切れない値となります。そこで、Ａモードと称し、公称入力範囲を５００００で除した分解能を採用したモードを準備しています。

表２－２Ａ１　入力範囲・分解能・正確度

公称入力範囲	範囲モード	実際の入力範囲 （±１０Ｖを越えた部分は無効）	分解能 mV/digit	正確度
±１０Ｖ	Ａ	－１３．１０７２Ｖ～＋１３．１０６８Ｖ	０．４	表２－２Ａ２参照
	Ｂ	－１０．０００Ｖ～＋９．９９９６９Ｖ	０．３０５	
±５Ｖ	Ａ	－６．５５３６Ｖ～＋６．５５３４Ｖ	０．２	
	Ｂ	－５．０００Ｖ～＋４．９９９８５Ｖ	０．１５３	
±２．５Ｖ	Ａ	－３．２７６８Ｖ～＋３．２７６７Ｖ	０．１	
	Ｂ	－２．５００Ｖ～＋２．４９９９２Ｖ	０．０７６	
０～＋１０Ｖ	Ａ	０Ｖ～＋１３．１０７２Ｖ	０．２	
	Ｂ	０Ｖ～＋９．９９９８５Ｖ	０．１５３	
０～＋５Ｖ	Ａ	０Ｖ～＋６．５５３５Ｖ	０．１	
	Ｂ	０Ｖ～＋４．９９９９２Ｖ	０．０７６	

表２－２Ａ２　スキャン速度と正確度

スキャン動作モード	単Ｃｈサンプル	複数Ｃｈサンプル	基本正確度
高速スキャンモード	２００ＫＨｚ	５×Ｃｈ（ｕＳ）	±０．０２３％ＦＳ
低速スキャンモード	２００ＫＨｚ	１０×Ｃｈ（ｕＳ）	±０．０２３％ＦＳ

表２－２Ｂ　ＡＤデータｖｓアナログ入力（Ａモード）

ＡＤデータ	アナログ入力範囲				
Hex/10 進	±１０Ｖ	±５Ｖ	±２．５Ｖ	０～＋１０Ｖ	０～＋５Ｖ
FFFF/65535	+13.1068	+6.5534	+3.2767	+13.1072	+6.5535
E1A8/57768	+10.0000	+5.0000	+2.5000		
C350/50000				+10.0000	+5.0000
8001/32769	+0.0004	+0.0002	+0.0001		
8000/32768	0.0000	0.0000	0.00000		
7FFF/32767	-0.0004	-0.0002	-0.0001		
61A8/25000				+5.0000	+2.50000
1E58/7768	-10.000	-5.0000	-2.50000		
0001/1	-13.1068	-6.5534	-3.2767	+0.0002	+0.0001
0000/0	-13.1072	-6.5536	-3.2768	0.0000	0.00000

表２－２Ｃ　ADデータｖｓアナログ入力（Bモード）

ADデータ	アナログ入力範囲				
Hex/10 進	±１０Ｖ	±５Ｖ	±２．５Ｖ	０～＋１０Ｖ	０～＋５Ｖ
FFFF/65535	+9.99969	+4.99985	+2.49992	+9.99985	+4.99992
E1A8/57768	+7.62939	+3.81470	+1.90735		
C350/50000				+7.62939	+3.81470
8001/32769	+0.00031	+0.00015	+0.00008		
8000/32768	0.00000	0.0000	0.00000	+5.00000	+2.50000
7FFF/32767	-0.00031	-0.00015	-0.00008		
61A8/25000				+3.81470	+1.90735
1E58/7768	-7.62939	-3.81470	-1.90735		
0001/1	-9.99969	-4.99985	-2.49992	+0.00015	+0.00008
0000/0	-10.00000	-5.00000	-2.50000	0.00000	0.00000

２－３　アナログ入力特性（誤差、ドリフト、雑音、保護対策）

ＡＤ変換誤差：　本ボードのＡＤ入力は±１０Ｖ範囲で調整されていますが、高精度部品の使用により入力範囲を切り替えても多くの用途では再調整の必要がありません。しかしながら特定の入力範囲で最も正確度を良くしたいときは再調整が必要です。
温度ドリフト：　ＡＤボードの周囲温度が変化した時、同一アナログ入力に対するＡＤ変換データが変化する度合いを示します。本ボードでは１０ｐｐｍ／℃typ となります。
経年変化：　経年変化のデータはありません。充分な精度の維持が必要な用途では年に１～２回程度標準電圧源などで校正し必要に応じて再調整を行うと良いと思われます。
内部雑音：　本ボードでは内部雑音は±４LSB となります。
入力耐圧：　本ボードでは±４０Ｖまでの過電圧に対して保護されていますが、これを超える入力電圧が印加されると入力回路素子の破壊による故障の原因となります。このような場合は、抵抗＋ツェナーダイオード等の保護回路が有効です。実際の定数を決定するには、ある程度のカットアンドトライが必要になると思われます。

２－４　外部サンプルホールド制御

本ボードに採用されているＡＤ変換器はサンプルホールド機能を備えてはいますが、複数チャンネルをサンプリングする場合には、アナログ入力をマルチプレクサで自動的に切り替え、サンプルホールド・ＡＤ変換を繰り返す逐次サンプル方式とて動作します。そのため、隣接チャンネル間では５ｕＳ（低速モードでは１０ｕＳ）の時間差が含まれる事になります。

この状況を改善するため、外部にサンプルホールドＩＣを用意して同時サンプリング動作を行わせる事ができるようにＴＴＬレベルの制御信号を用意しました。この信号は、ボード上のジャンパーを接続する事によって外部に供給する事ができるようになっています。

２－５　デジタル入出力回路

外部トリガ入力、外部割込み入力、汎用１ビットデジタル入力は全てＴＴＬレベルであり、夫々１０ＫΩで、プルアップされています。クロック出力もＴＴＬレベルです。

汎用１ビットデジタル出力はオープンコレクタですが、接続先がＴＴＬレベルの時は、ボード上のジャンパーＳ－ＰＵＰにより１ＫΩのプルアップ抵抗を接続して使用する事ができます。（出荷時設定）

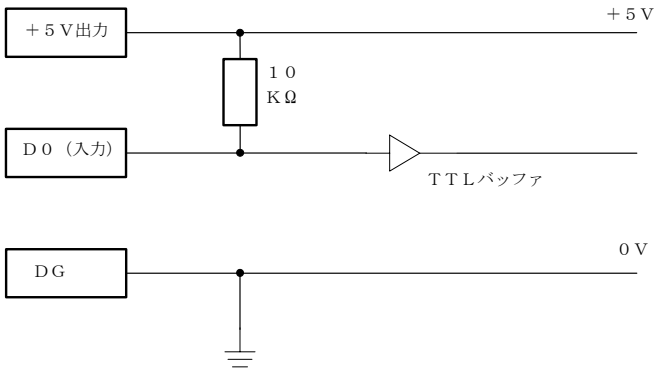


図２－４Ａ　デジタル入力

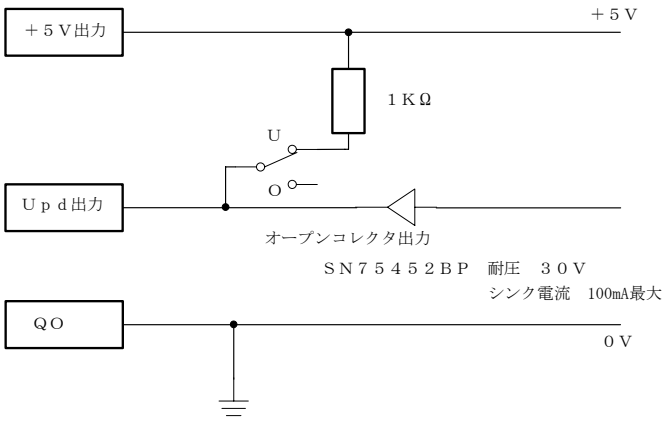


図２－４Ｂ　汎用１ビット出力

第３章　制御・操作

３－１　ＡＤサンプリング動作、トリガ動作の様子

本ボードには大別して３種類のサンプリングモードがあります。

第一に挙げるモードは有限自動サンプリングモードです。このモードは、有限データ数のサンプリング終了と同時にサンプリングモードそのものを停止する機能です。また、このモードでは、サンプリングされたアナログ変換データをアプリケーションに引き渡すため、専用ハンドラー内部に予め確保される DLL バッファー、或いはアプリケーション内部にユーザーが確保するユーザーバッファーの何れかを指定して使用します。

次に挙げるモードは無限自動サンプリングモードです。このモードは、DLL バッファーをリングバッファ形式として運用し、強制的にサンプリングを停止するまでの間連続的にサンプリングを行います。

最後に挙げるモードはマニュアルサンプリングモードです。このモードでは、あらかじめ設定されたチャンネル群に対し、１サイクルのサンプリングを行います。このモードでは、ハンドラー内部に小容量のバッファーを用意し、そのバッファーを経由してサンプリングされたアナログ変換データの受け渡しを行います。

有限自動サンプリングモード

このモードでは、サンプリングパラメータを設定する際、チャンネル当たりサンプリング数を内部レジスタにセットし、更にトリガモードレジスタのビット１（有限サンプリング機能ビット）をセットします。
この状態でサンプリングを開始すると、ハンドラーによりサンプリング回数がモニタリングされ、残りカウント数が０になった事を検出し、ハンドラーによりサンプリング動作が停止されます。

無限自動サンプリングモード

このモードでは、サンプリングパラメータを設定する際、チャンネル当たりサンプリング数のセットは行わずトリガモードレジスタのビット１（有限サンプリング機能ビット）もセットしません。
この状態でサンプリングを開始すると、DLL バッファーをリングバッファとして扱い連続サンプリングを開始します。この場合にサンプリングを停止するためには、明示的にサンプリングストップを指示する必要があります。

マニュアルサンプリングモード

このモードでは、サンプリングパラメータを設定し、その後ソフトウェア上でマニュアルサンプリング開始コマンドを発行することで、１サイクルのサンプリングを行います。このモードは、通常、サンプリング動作のデバッグモード或いは、サンプリング動作そのものに時間的要因が含まれない動作の場合に使用されます。

３－２　オンボードメモリとユーザーアプリケーションとの関係性

本ボードには、通常の FIFO メモリではなく汎用の SDRAM を FPGA で制御する事による擬似的な FIFO メモリが実装されています。この事により

１　通常の FIFO メモリに対しはるかに大容量のメモリを使用できる
２　FIFO メモリ内部動作が FPGA によって実現されているため、この擬似メモリの内部動作が本ボード経由でアプリケーション側からも把握でき、その結果より細かい制御が可能となる

というような利点を挙げる事ができます。

また、次ページに AD 変換ボードハードウェアと、デバイスドライバー、ユーザーアプリケーション等のソフトウェアとの関連性を纏めた形で記載しています。

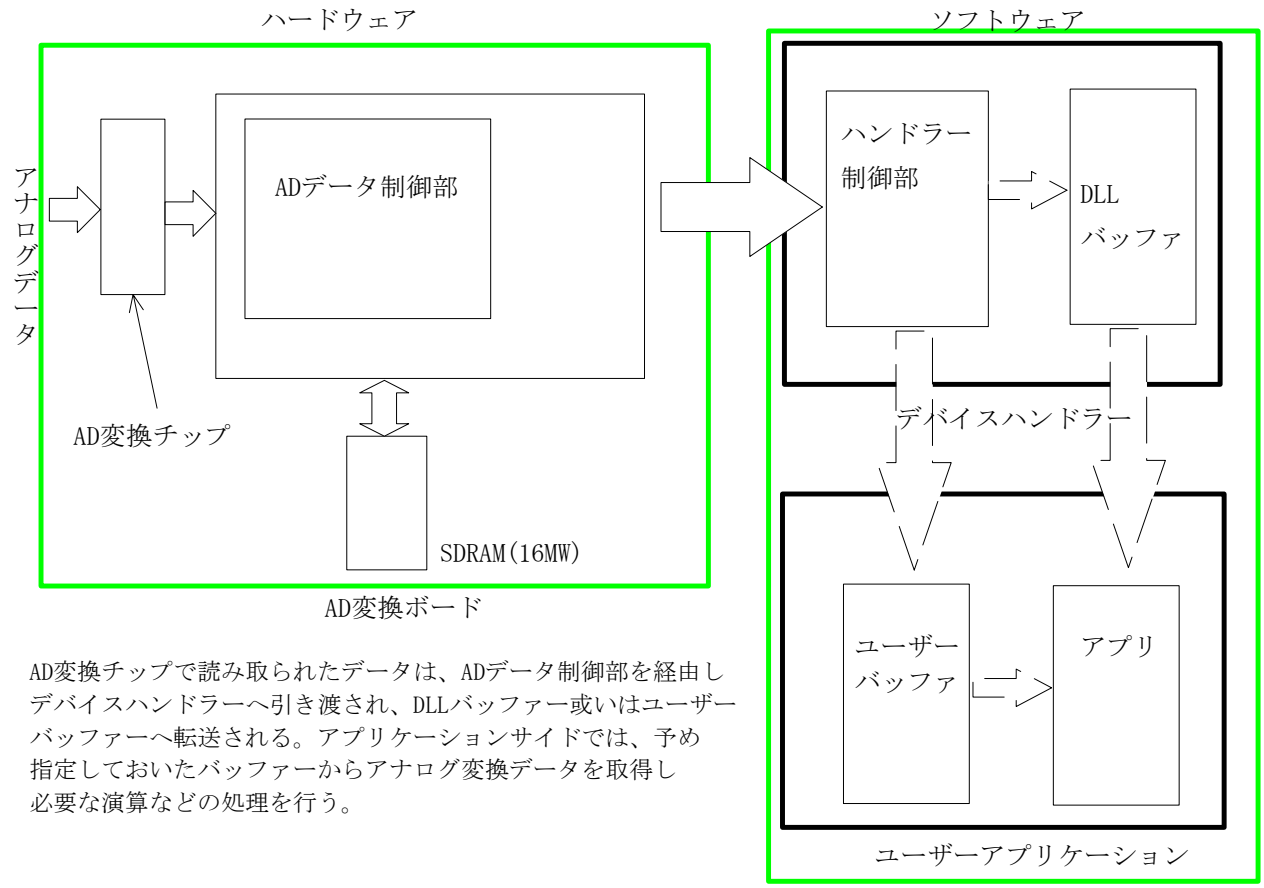


図３－２　ハードウェアとソフトウェアの相互関係図

上の図で示すように、ＡＤ変換ボードで得られたアナログ変換値は、デバイスドライバーを経由して、同ドライバー内部に確保される DLL バッファー、或いはユーザーアプリケーション側で用意するユーザーバッファー何れかに転送されます。ユーザーアプリケーション側では、予め指定しておいたバッファーから変換データを取り出し必要な処理を行うことになります。

３－３　制御レジスタアドレスマップ

表３－３に本ボードのアドレスマップを示します。
表中【ＢＡＳＥ】で示される数値は、プラグアンドプレイで割り付けられるＩＯアドレスのベース値です。

表３－３　制御レジスタのＩ／Ｏアドレス

Ｉ／Ｏアドレス(Hex)	ＩＮ／ＯＵＴ	ポート／レジスタ名・機能	参照
【ＢＡＳＥ】＋Ｆ	ＯＵＴ	マルチサンプリングクロック源　分周比設定	３－25 項
	ＩＮ		
【ＢＡＳＥ】＋Ｅ	ＯＵＴ	マルチサンプリングクロック源　モード選択	３－24 項
	ＩＮ		
【ＢＡＳＥ】＋Ｄ	ＯＵＴ	マルチサンプリング回数指定（初期値＝１）	３－23 項
	ＩＮ		
【ＢＡＳＥ】＋Ｃ	ＯＵＴ	外付け・同時サンプルホールド制御（オプション）	３－22 項
	ＩＮ	アナログ回路設定状態読出し（注１）	３－５項
【ＢＡＳＥ】＋Ｂ	ＯＵＴ	アナログ回路接続設定（注１）	３－５項
	ＩＮ	ボード番号（ロータリースイッチ設定値）	３－４項
【ＢＡＳＥ】＋Ａ	ＯＵＴ	汎用デジタル出力（ラッチ）	３－21 項
	ＩＮ	汎用デジタル入力（現在値）	３－20 項
【ＢＡＳＥ】＋９	ＯＵＴ	クロック源分周比設定	３－９項
	ＩＮ		
【ＢＡＳＥ】＋８	ＯＵＴ	クロック源選択	３－８項
	ＩＮ	ＦＩＦＯメモリ保持データ数読込み（注１）	３－15 項
【ＢＡＳＥ】＋７	ＯＵＴ	アナログトリガレベル（２）設定	３－10 項
	ＩＮ	ボード制御部リセット	３－４項
【ＢＡＳＥ】＋６	ＯＵＴ	アナログトリガレベル（１）設定	３－10 項
	ＩＮ	マニュアルサンプリングスキャン	３－12 項
【ＢＡＳＥ】＋５	ＯＵＴ	ステータス（ビット指定）クリア	３－14 項
	ＩＮ	ステータス取得	３－13 項
【ＢＡＳＥ】＋４	ＯＵＴ	割り込み制御（要因設定）	３－17 項
	ＩＮ		
【ＢＡＳＥ】＋３	ＯＵＴ	トリガモード指定（ソフトトリガ実行を含む）	３－11 項
	ＩＮ		
【ＢＡＳＥ】＋２	ＯＵＴ	データコード指定、スキャン速度指定	３－６項
	ＩＮ		
【ＢＡＳＥ】＋１	ＯＵＴ	割り込み要求クリア	３－18 項
	ＩＮ		
【ＢＡＳＥ】＋０	ＯＵＴ	アナログ入力スキャン最終チャンネル指定	３－７項
	ＩＮ	ＡＤデータ読出し	３－16 項

ＩＮ／ＯＵＴはパソコン側から見た方向
全てのポートはバイトアクセス
（注１）：これらの機能は、FCAD516PCI で追加導入されている為、マイクロサイエンス製の専用ドライバー（H686z_2K.Dll）では使用できません。（汎用 IO ドライバーでは使用可能です）

３－４　ボード制御部リセット

　ｒｓｔ＝ｉｎｐ（ＢＡＳＥ＋０ｘ７）；　　//　制御部リセット操作

本ボード制御部全体を初期化します。この操作で読み込まれた値は本ボードのＩＤです。本操作は電源ＯＮ又はパソコン本体のハードウェアリセットと同等の機能ですが、汎用デジタル出力だけは変化せず保持されます。
本操作により

- 本ボード上の各制御レジスタを初期化する。
- ボードステータスを初期化する。
- サンプリング中であればこれを中止する。
- ＦＩＦＯメモリをクリアする。（格納された読出し待ちＡＤデータは失われる）

尚、クロック源及び分周比は無効となるため再設定が必要となります。

表３－４Ａ【ＢＡＳＥ】＋０ｘ７Ｈ入力ポートの構成

ビット	各ビットの機能・意味
Ｂ７～Ｂ０	０ｘ０ＦＨ

【※】ここで読み込まれるボードＩＤはＰＣＩバス上のデバイスＩＤとは全く無関係です。

本ボードを複数使用する場合
本ボードのＩＯアドレスはプラグアンドプレイによりダイナミックに割り当てられる為、複数のボードを使用する場合ソフトウェア側から夫々のボードを特定する手段が必要になります。（この手段がないと、例えばハードウェア構成が変化した場合など、ソフトウェア側からボードの特定を行う事が困難になります。）
そのために、このスイッチが用意されています。
また、専用ハンドラーを使用する際には、例えボード１枚のみの使用であってもこのスイッチを“０”に設定しておかなければなりません。

　ＢＮ＝ｉｎｐ（ＢＡＳＥ＋０ｘＢ）；　　//　ボード番号設定スイッチの値を読み込む

表３－４Ｂ【ＢＡＳＥ】＋０ｘＢＨ入力ポートの構成

ビット	各ビットの機能・意味
B7～B4	未使用
B3 B2 B1 B0	ボード番号設定スイッチの値（０～ＦＨ）

３－５　アナログ回路接続設定・読出し

本ボードでは、ボード内部のアナログ回路設定を半導体スイッチによりソフトウェア的に切り替える事ができます。

ｒｅｔ＝ｉｎｐ（ＢＡＳＥ＋０ｘＣ）；　　// 設定値読み込みの場合
ｏｕｔｐ（ＢＡＳＥ＋０ｘＢ、Ｓｅｔ＿Ｄａｔａ）；　　// 設定値書き込みの場合
（Ｓｅｔ＿Ｄａｔａはレジスタに書き込む内容）

切り替えられる機能は、以下の通りとなります。

- 入力レンジ（２．５Ｖ、５Ｖ、１０Ｖ）
- 感度切り替え（Ａレンジ／Ｂレンジ）
- 入力モード切り替え（バイポーラ／ユニポーラ）
- 入力接続切り替え（シングルエンド／差動入力）

これらについて実際の組み合わせを表３－５Ａ～Ｂに示します。

表３－５Ａ　アナログ回路接続設定組み合わせ（１６Ｂｉｔ）

入力レンジ	Ａレンジ	入力感度	Ｂレンジ	入力感度
±２．５Ｖ	-3.2768V～+3.2767V	0.1mV/Digit	-2.5Ｖ～+2.49992Ｖ	0.076mV/Digit
±５Ｖ	-6.5536V～+6.5534V	0.2mV/Digit	－5.0V～＋4.99985V	0.153mV/Digit
±１０Ｖ	-13.1072～+13.1068V	0.4mV/Digit	-10V～＋9.99969V	0.305mV/Digit

表３－５Ｂに実際の回路との間で取り決められている出力ビット割り当てを示します。

表３－５Ｂ　【ＢＡＳＥ＋BH】出力ポートの構成

ビット	機能	“1”の時	“0”の時
B7～B5	未使用		
B4	入力接続切り替え	シングルエンド	差動入力
B3,B2	入力感度切り替え	B3B2=00	５Ｖ
		B3B2=01	１０Ｖ
		B3B2=10	２０Ｖ
		B3B2=11	使用不可
B1	ユニポーラ／バイポーラ切り替え	ユニポーラ	バイポーラ
B0	Ａレンジ／Ｂレンジ切り替え	Ｂレンジ	Ａレンジ

表３－５Ｃには、回路設定条件読み出しポートの入力ビット割り当てを示します。

表３－５Ｃ　【ＢＡＳＥ＋CH】入力ポートの構成

ビット	機能	“１”の時	“０”の時
B7～B5	未使用		
B4	入力接続切り替え	シングルエンド	差動入力
B3,B2	入力感度切り替え	B3B2=00	５Ｖ
		B3B2=01	１０Ｖ
		B3B2=10	２０Ｖ
		B3B2=11	使用不可
B1	ユニポーラ／バイポーラ切り替え	ユニポーラ	バイポーラ
B0	Ａレンジ／Ｂレンジ切り替え	Ｂレンジ	Ａレンジ

実際の所、これらの組み合わせ全てが使用可能な訳ではなく、以下の組み合わせのみが有効となります。

表３－５Ｄ　有効な設定組み合わせ一覧

		B1=1（ユニポーラ）		B1=0（バイポーラ）	
ビット３，２	入力感度	Ａレンジ	Ｂレンジ	Ａレンジ	Ｂレンジ
B3=0,B2=0	５Ｖ	０～＋５Ｖ	０～＋５Ｖ	±２．５Ｖ	±２．５Ｖ
B3=0,B2=1	１０Ｖ	０～＋１０Ｖ	０～＋１０Ｖ	±５Ｖ	±５Ｖ
B3=1,B2=0	２０Ｖ	該当なし	該当なし	±１０Ｖ	±１０Ｖ

また、これら全ての有効な組み合わせについて、差動入力・シングルエンド入力の切り替えが可能です。

専用ハンドラーでは、上表の有効な組み合わせのみを設定し、それ以外についてはエラーを返します。
また、以下に実際に設定を行う例を示します。

入力レンジ：±５Ｖ（バイポーラ）
レンジ指定：Ｂレンジ
差動入力接続
の場合、

B4=0
B3,B2=01
B1=0
B0=1 となるので、これらを組み合わせた値、0x05 がこのレジスタに書き込まれるべき数値ということになります。しかし、実際には予めご使用いただく設定がハードコーディングされておりますので、この数値を直接出力しても所要のセッティングにはなりません。また、通常使用される場合にはこのレジスタを書き換える可能性はかなり低いものと思われます。実際に内容を変更するには、以下の手続きにより行います。

```
1　outp（BASE+0xB,0）；　　// まず、このレジスタの内容を初期化します。
2　retc =(BASE+0xC);　　// 初期状態に戻したレジスタの内容を取得します。
3　retc = retc^OutPutData;　　// 出力すべきデータとの間で排他的論理和を計算します。
4　outp (BASE+0xB,retc);　　// 計算結果をレジスタに書き込みます。
```

尚、ハードコーディングされた設定以外の値とする場合には、ステップ１のレジスタ初期化を行わないと所要の設定ができないため注意が必要です。

３－６　データコード指定

ｏｕｔｐｗ（ＢＡＳＥ＋０ｘ２、ｄｃｄ）；
AD データコード（バイナリ、または２の補数）の設定データ及びサンプリングスキャン速度指定データを書き込みます。

表３－６Ａ　【ＢＡＳＥ＋２H】出力ポートの構成

ビット	各ビットの機能・意味	＝１の時	＝０の時	リセット時
B7～B5	未使用			
B4	AD データコード	２の補数	バイナリ	0
B3～B1	未使用			
B0	サンプリングスキャン速度指定	高速（5uS）	低速(10uS)	0

３－７　アナログ入力スキャン最終チャンネル指定

ｏｕｔｐ（ＢＡＳＥ＋０ｘ０、ｅｎｄ＿ｃｈ）；
サンプリングスキャン動作は、チャンネル０から、上の式で示される最終チャンネルまで連続して行います。
即ち、サンプリングチャンネル数はｅｎｄ＿ｃｈ＋１となります。

表３－７Ａ　【ＢＡＳＥ＋０H】出力ポートの構成

ビット	各ビットの機能・意味	設定値	リセット時
B7～B5	未使用		
B4～B0	スキャン最終チャンネル	0H～FH	0

３－８　クロック源選択

ｏｕｔｐ（ＢＡＳＥ＋０ｘ８、ｃｌｋ）；　// ｃｌｋ：クロック源選択データ

ここで選択したクロック源信号は次項で説明するプログラマブルカウンタにより分周されて所要のサンプリングクロックになります。例えば、外部クロック源入力を選択し、分周比を１に設定すれば、外部イベントに同期したサンプリングを行うことができます。

表３－８　【ＢＡＳＥ＋８Ｈ】出力ポートの構成

ビット	各ビットの機能・意味	＝１の時	＝０の時	リセット時
B7	外部クロック源の有効極性指定	↑（＋）	↓（－）	0
B6～B5	内部クロック源の選択データ	B6B5=11	無効	００
		B6B5=10	オプション	
		B6B5=01	８．１９２MHｚ	
		B6B5=00	１０．０００MHｚ	
B4	クロック源選択（外部／内部）	外部	内部	0
B3～B0	未使用			

オプションのクロックは本基板上 OSC4 に追加装着することで使用可能となります。（１０MHｚ以下）
対応可能発振器は SG-8002DC-□□□M-PTBS 型（セイコーエプソン）が入手可能です。

３－９　クロック源分周比設定

ｏｕｔｐ（ＢＡＳＥ＋０ｘ９、ＤｉｖＬ）；　// 分周比Ａ（下位バイト）
ｏｕｔｐ（ＢＡＳＥ＋０ｘ９、ＤｉｖＬＨ）；　//上位バイト
ｏｕｔｐ（ＢＡＳＥ＋０ｘ９、ＤｉｖＨ）；　// 分周比Ｂ（下位バイト）
ｏｕｔｐ（ＢＡＳＥ＋０ｘ９、ＤｉｖＨＨ）；　//上位バイト

クロック源の分周比を所要のレジスタに書き込みます。必ず下位、上位の順に連続して書き込む必要があります。
実際の分周は 16 ビットのカウンターを 2 段カスケード接続したカウンターによって行われます。
内部クロック源１０MHｚを選択した時の最長クロック周期は
１００ｎＳ×（０ｘＦＦＦＦ×０ｘＦＦＦＦ）＝０．１ｕＳ×4294836225＝429.4836225 秒
となります。

３－10　アナログトリガレベル設定

ｏｕｔｐ（ＢＡＳＥ＋０ｘ６、ＴｒｇＬ）；　// ＴｒｇＬ＝トリガレベルデータ１
ｏｕｔｐ（ＢＡＳＥ＋０ｘ７、ＴｒｇＨ）；　// ＴｒｇＨ＝トリガレベルデータ２

アナログトリガレベル指定データをそれぞれのアドレスに書き込みます。設定にあたっては、必ず両方のデータを書き込む必要があります。
（トリガモードによってはＴｒｇＨがダミーデータとなりますが書込み操作そのものは必要です。）
アナログトリガ待ち状態では、チャンネル０入力が 5uS 周期で連続して監視サンプリングされ AD 変換データの上位８ビットが上記で設定されたトリガレベルと比較されています。トリガが認識されると３－１０項で示した分周動作が開始され、連続サンプリングが始まります。
ここで両トリガレベルと次項で説明するトリガモードとの間には、表３－１０Ａで説明される関係があります。

表３－１０Ａ　トリガモード対トリガレベルデータ

トリガレベルデータ	エッジトリガの場合	レベルトリガの場合	レンジトリガの場合
ＴｒｇＬ	トリガレベル(注)	トリガレベル	トリガレベル下限値
ＴｒｇＨ	トリガレベル(注)	ダミーデータ	トリガレベル上限値

注　エッジトリガの場合、ＴｒｇＨ－ＴｒｇＬの値がヒステリシスとなります。即ち立ち上がりエッジの場合はＴｒｇＬ未満からＴｒｇＨを超えた時点で、また立ち下がりエッジの場合はＴｒｇＨ以上からＴｒｇＬを下回った時点でトリガ成立となります。

表３－１０Ｂ　各アナログ入力範囲に対するスパン、トリガレベル分解能

アナログ入力範囲	Ａレンジの場合		Ｂレンジの場合	
	Vspan	分解能（Vspan/256）	Vspan	分解能（Vspan/256）
±10V	26.2114V	102mV	20V	78.125mV
±5V	13.1072V	51mV	10V	39.0625mV
±2.5V	6.5536V	25.5mV	5V	19.5313mV

３－11　トリガモード設定（ソフトトリガ実行を含む）

ｏｕｔｐ（ＢＡＳＥ＋０ｘ３、TrgMode）；
トリガモード及びサンプリングモードデータを書き込みます。

表３－１１【ＢＡＳＥ＋０ｘ３】出力ポートの構成

ビット	各ビットの機能・意味	＝１の時	＝０の時	リセット時
B7	ソフトトリガ制御	発生	禁止（強制停止）	0
B6	外部トリガ入力信号制御	許可	禁止（強制停止）	0
B5	内部（アナログ）トリガ制御	許可	禁止（強制停止）	0
B4	トリガ極性選択	↑（+）	↓（−）	0
B3	トリガ認識モード選択２	エッジ	レベル	0
B2	トリガ認識選択モード１	レンジ	レンジ以外	0
B1	未使用			0
B0	サンプリングモード選択	連続サンプリング	マニュアルサンプリング	0

B7：ソフトトリガ制御ビットです。このビットをセットすると連続サンプリング開始と同時にサンプリングが開始されます。
B6：外部トリガ入力信号をサンプリングトリガとして使用するかどうかを指定するビットです。このビットをセットすると外部トリガ入力信号が B4,B3 で指定したモード条件に一致した時点でサンプリングが開始されます。
B5：アナログトリガ信号をサンプリング開始条件とするかどうかを指定するビットです。このビットをセットすると、アナログ入力信号が B4,B3 で指定される条件になった時点でサンプリングが開始されます。この条件には前項で説明したトリガレベルも関係します。
B4：B6,B5 で指定するトリガ信号の有効エッジ若しくは有効レベルについて立ち上がりエッジ／立下りエッジ若しくはハイレベル／ローレベルいずれを選択するかを指定するビットです。
B3：B6,B5 で指定するトリガ信号についてエッジ検出かレベル検出かを指定するビットです。
B2：B6,B5 で指定するトリガ信号についてその扱いをレンジモードとするかどうかを指定するビットです。
B0：サンプリング動作が連続サンプリングかマニュアルサンプリングかを指定するビットです。

３－12　マニュアルサンプリングスキャン

ｒｅｔ＝ｉｎｐ（ＢＡＳＥ＋０ｘ６）；　// ｒｅｔ：ダミーデータ
本操作により予め設定されたチャンネル群に対し、１サイクルのサンプリングを行います。変数ｒｅｔには特に意味は無く、この操作を行ったときのボード内部の動作によりサンプリングを行います。
サンプリングされたアナログデータは、通常のサンプリングと同様 FIFO メモリに書き込まれますので、アプリケーション側からは、同じように FIFO メモリからの読出しを行うことでアナログデータを得る事ができます。

３－13　ステータス入力

ｓｔｓ＝ｉｎｐ（ＢＡＳＥ＋０ｘ５）；
本ボードのステータスデータを読み込みます。

表３－１３【ＢＡＳＥ＋５H】入力ポートの構成

ビット	各ビットの機能・意味	＝１の時	＝０の時	リセット時
B7	EOS : 各回サンプリングスキャン終了　【注１】	終了済み	実行中、他	0
B6	INT : 割り込み要求発生　【注１】	発生済み	未発生、他	0
B5	TIM : 各回サンプリングスキャン開始　【注１】	開始済み	未開始、他	0
B4	TGD : トリガ発生認識　【注２】	発生済み	未発生	0
B3	EOM : 各回マルチサンプリング終了　【注１】	終了済み	実行中、その他	0
B2	ERR : エラーフラグ　【注１】	発生	未発生	0
B1	HLF : Not Half Full　【注２】	１／２未満	１／２以上	1
B0	EMP : Not Empty　【注２】	データあり	データなし	0

【注１】一旦セットされるとクリアされるまで状態を保持するラッチフラグ
【注２】現在の状態を常に反映する状態フラグ

各ビットの機能・意味詳細説明
EOS :　各回のサンプリングスキャンが終了した時点でセットされるフラグ
本ボード用ハンドラーでは、このビットを参照する事で、１スキャン完了のタイミングを、メッセージによりユーザーに通知する事ができる
INT :　所要の割り込み条件が成立し、割り込み信号が出力された時点でセットされるフラグ
TIM :　各回のサンプリングスキャンが開始される度に、セットされるフラグ
TGD :　所要のトリガ条件が成立するとセットされる。またトリガ禁止操作でクリアされる。
EOM :　各回のマルチサンプリングが終了するたびにセットされる。マルチサンプリング機能を連続測定のサイクルとして利用する場合は各測定サイクルの終了フラグとして利用できる
ERR :　サンプリング動作時にエラーが発生するとセットされるフラグ、制御部クリアまで保持される。
セット要因は
１　FIFO メモリが満杯の状態で更に書き込みが行われた場合（データロストエラー）
２　最高サンプリング速度を上回る速度でサンプリングが行われた場合（オーバーランエラー）
HLF :　FIFO メモリに対する所要データ数の書き込みが行われた時点でクリアされるフラグ
このフラグがクリアされていれば、予めセットされたデータ数が FIFO メモリに格納されている事が保証されている。
EMP :　FIFO メモリに１ワード以上のデータがある時にセットされるフラグ、アプリケーション上は低速でサンプリングを行うケース以外ではほとんど使用されることはない。

３－14　ステータスクリア操作

ｏｕｔｐ（ＢＡＳＥ＋０ｘ５、ＳｔａｔｕｓＣｌｅａｒ）；　// StatusClear はレジスタクリアパターン
ステータスレジスタの各ビットをクリアします。クリアするビットとの対応は表３-13Ｂによります。

表３－14【ＢＡＳＥ＋５H】出力ポートの構成

ビット	各ビットの機能・意味	＝１の時	＝０の時	リセット時
B7	EOS :【注１】ビットのクリア	クリアする	クリアしない	0
B6	INT :【注１】ビットのクリア	クリアする	クリアしない	0
B5	TIM :【注１】ビットのクリア	クリアする	クリアしない	0
B4				
B3	EOM :【注１】ビットのクリア制御	クリアする	クリアしない	0
B2	ERR :【注１】ビットのクリア	クリアする	クリアしない	0
B1				
B0	FIFO メモリのクリア【注３】	クリアする	クリアしない	0

【注３】FIFO メモリ内容を全て破棄しフラグ出力も初期状態に戻す

３－15　ＦＩＦＯメモリ保持データ数読込み

ｍｅｍ＿ｎｕｍｂｅｒ＝ｉｎｐ（ＢＡＳＥ＋０ｘ８）；　// 保持データ数最下位バイト読込み
ｆｏｒ（ｌｏｏｐ＝０；ｌｏｏｐ＜３；ｌｏｏｐ＋＋）｛
ｍｅｍ＿ｎｕｍｂｅｒ｜＝（ｉｎｐ（ＢＡＳＥ＋０ｘ８）＜＜（８＊ｌｏｏｐ））；// 同上位ワードを加算

本ボード内部では、常にＦＩＦＯメモリに格納されているデータ数を更新しています。更新単位はワードなので上記ポートから読み出した数値は、そのまま内部ＦＩＦＯに格納されているデータ数と一致します。更に内部出力バッファーには、次の読出しコマンドに供えてほとんど常に１バイトのデータが用意されていますが、ステータスレジスタのビット０がセットされていた場合、読み出した値を＋１するかどうかは微妙な関係になります。（＋１するためには２バイトのデータがある事が必要だが、この条件は満足していないため）また、この機能もマイクロサイエンス製のボードでは使用できません。

３－16　ＡＤデータ読出し

ＡｄＤａｔａ＝ｉｎｐ（ＢＡＳＥ＋０ｘ０）；　// ＡｄＤａｔａは読み出したAD変換データ
ＡｄＤａｔａ｜＝（ｉｎｐ（ＢＡＳＥ＋０ｘ０）　＜＜　８）；
AD変換されたアナログデータはボード内のＦＩＦＯメモリに格納されており、アプリケーション側からはオフセット０からの連続した２バイトの読出しによってデータを引き取ります。仕組み的には、ＣＰＵのブロック転送命令を使用する事も不可能ではありませんが、Windows７以降のＯＳではハンドラー側で対応する必要があります。

３－17　割り込み制御（要因設定）

ｏｕｔｐ（ＢＡＳＥ＋０ｘ４、ＩｎｔＭａｓｋ）；　// 割り込み要因制御パターン：ＩｎｔＭａｓｋ

表３－１７【BASE＋０ｘ４】出力ポートの構成

ビット	各ビットの機能・意味	＝１の時	＝０の時	リセット時
B7	外部割込み信号の有効極性指定	↑（+）	↓（−）	0
B6	HalfFull 状態になった事による割り込み	許可	禁止	0
B5	NotEmpty 状態になった事による割り込み	許可	禁止	0
B4	各回マルチサンプリング終了による割り込み	許可	禁止	0
B3	１回サンプリングスキャン終了による割り込み	許可	禁止	0
B2	トリガ発生による割り込み	許可	禁止	0
B1	外部割込み信号による割り込み	許可	禁止	0
B0	連続サンプリングクロックによる割り込み	許可	禁止	0

B7：外部割込み信号が許可された場合の信号エッジ極性（↑↓）指定
B6：FIFO メモリ内の取り込みデータがハーフサイズを超えた状態になったタイミングによる割り込み制御
B5：FIFO メモリが空の状態から１データ取り込んだタイミングによる割り込み制御
B4：各回マルチサンプリング終了タイミングによる割り込み制御
B3：各回サンプリングスキャン終了タイミングによる割り込み制御
B2：トリガ発生による割り込み制御
B1：外部割込み信号による割り込み制御
B0：指定クロックの有効エッジによる割り込み制御

３－18　割り込み要求クリア

ｏｕｔｐ（ＢＡＳＥ＋０ｘ１、１）；　// 割り込み要求信号出力許可
ｏｕｔｐ（ＢＡＳＥ＋０ｘ１、０）；　// 割り込み要求信号クリア（出力禁止）
本ボードからの割り込み要求信号はソフトウェア上（デバイスドライバー内）でクリアする必要があります。PCI バスの割り込み信号がクリア操作まで割り込み要求状態を保持するレベル動作であるためです。
またこのポートはラッチポートですので、割り込み処理ルーチンの先頭で、ビット０を使用して一旦クリア操作を行うと、その状態は保持され次の割り込み要求は出力することができません。これを解消するには、割り込み処理を起動した割り込み要求がクリアされた後、再度ビット０をセットする操作が必要です。この一連の操作はデバイスドライバー内部で行われます。

表３－１８【BASE＋１H】出力ポートの構成

ビット	各ビットの機能・意味	＝１の時	＝０の時	リセット時
B7～B1	未使用			
B0	PCI バスへの割り込み信号出力制御	出力許可	出力禁止	0

３－19　マススタースレーブ接続

本ボードを複数枚使用して同期運転することができます。この場合、トリガを検出するボードをマスター、他のボードをスレーブとします。スレーブはマスターからのクロックを受けて同期運転を行いますが、サンプリングタイミングに最大 125nS の遅れが生じます。
マススタースレーブ間の接続及び操作については次の通りとなります。

ボード上の設定

マススタースレーブの棲み分けは、ボード上にあるロータリースイッチ（SW-BN）によって行う。マスターボードでは、スイッチの設定は“０”とし、スレーブボードでは非“０”とする。ハンドラーからはスイッチの値が“０”であるマスターが第一番に、以下“１”であるスレーブ、“２”であるスレーブという順番で認識され扱うアナログデータについてもこの順番によって処理される。

ボード間の接続

マスターボードのクロック出力をスレーブボードのクロック入力に接続する。マスターボードについては、全てのトリガ方式を使用することができる。

ソフトウェア設定

スレーブボードは全てクロック源を外部立下りとし、クロック源分周比も１／１（非分周）と設定、トリガーソースについては、ソフトウェアトリガのみとしアナログトリガ、外部トリガは禁止しておく。
連続サンプリング動作開始は、最初に（ソフトウェアトリガで）スレーブボードに対して行い、最後にマスターボードを任意のトリガ条件で起動する。
【注】これらソフトウェア設定の記述は汎用ハンドラーを用いてアプリケーションをくみ上げる際の指標として記載したものです。専用ハンドラーでは、これらの仕組みはハンドラー内部で処理されているためアプリケーション側からは見ることができません。

マルチサンプリングとの関係

マルチサンプリング機能と、マススタースレーブ接続との関係は多少複雑になります。要点としては、マルチサンプリング用クロック源を内部（１０MHｚ）とするか、外部（汎用入力端子）とするかという点に絞られます。
サンプリングクロックについては、スレーブボードでは、専用の設定（外部クロック立ち下がりエッジ１分周）となりますが、マルチサンプリング用クロックについては、マスター、スレーブ共に同一の設定となります。
即ち、外部クロック源の場合、マスター、スレーブ共に共通のクロック源を汎用入力端子へ接続する事になります。またマルチサンプリング機能関係の設定（マルチサンプリング回数、ハード平均動作の有無、クロック分周比）についても、マスター、スレーブ共に同一の設定となります。

３－20　汎用デジタル入力

Ｄｉｎ＝ｉｎｐ（ＢＡＳＥ＋０ｘＡ）；　// 汎用１ビット TTL 入力（現在値）

表３－２０【ＢＡＳＥ＋AH】入力ポートの構成

ビット	各ビットの機能・意味	＝１の時	＝０の時
B7～B1	未使用		
B0	Ｉ０：汎用デジタル入力ビット０	Ｈｉｇｈ（開放）	Ｌｏｗ（０Ｖレベル）

本入力ポートについては、外部クロック使用の場合マルチサンプリング機能専用クロック入力としても使用されます。

３－21　汎用デジタル出力

ｏｕｔｐ（ＢＡＳＥ＋０ｘＡ、Ｄｏｕｔ）；　// 汎用１ビットラッチ出力

表３－２１【ＢＡＳＥ+AH】出力ポートの構成

ビット	各ビットの機能・意味	＝１の時	＝０の時	リセット時
B7　～　B1	未使用			
B0	Ｑ０：汎用デジタル出力ビット０	Ｈｉｇｈ	Ｌｏｗ	0

表中の値はＳ－ＰＯＬジャンパーが“P”側に設定されている場合です。

【注３】出力素子はＳＮ７５４５２ＢＰを使用しています。（耐圧＝３０Ｖ、出力電流３００ｍAmax デバイスそのものは出力電流３００ｍＡまで使用する事が可能ですが、その時の出力電圧がＴＴＬローレベルの許容範囲を超えてしまうため１００ｍＡまでとします。）

【注４】電源投入またはハードウェアリセット直後の汎用デジタル出力の状態は“０”ですが、ソフト的な制御部リセット操作ではその状態は変わりません。

【注５】汎用デジタル出力の論理はボード上のジャンパーＳ－ＰＯＬによって任意に設定することができます。また出力回路については、ボード上のジャンパーＳ－ＰＵＰによってオープンコレクタ（O 側）または１KΩのプルアップ（U側：標準）を選択できます。

３－22　外付け同時サンプルホールド制御

本ボードでは、オフセット０ｘ０Ｃへの出力により外付け同時サンプルホールド回路制御のタイミングを調整する事ができます。通常、サンプリング開始と共にアナログ入力はホールド機能によりラッチされアナログ変換が行われますが、外付けのサンプルホールド回路を使用する事で、スキャンタイミング毎に全チャンネルのアナログ値をサンプルホールドし同時サンプリングを行う事が可能です。しかしその際、外付けサンプルホールド回路の出力が安定するまでに有限の時間が必要で（通常１ｕＳ程度）それまでの間アナログ値が多少とも変動します。本機能は、この有限時間を先頭チャンネルのアナログ変換に先立って挿入するもので、この機能により、全チャンネルにおいてアナログ変換誤差のばらつきが抑えられます。その一方、この機能を有効にする事により、若干ではありますが、サンプリング速度が低下するため、この点での注意も必要です。

表２２【ＢＡＳＥ＋０ｘ０Ｃ】出力ポートの構成

ビット	各ビットの機能・意味	＝１の時	＝０の時	リセット時
B7　～　B1	未使用			
B0	サンプルホールド機能有効・無効	有効	無効	0

３－23　マルチサンプリング回数指定

マルチサンプリング動作を使用する際、通常のサンプリングクロック毎にサンプリングスキャン（指定チャンネル群に対して各一回AD変換を行う動作）を実行する回数を指定します。このレジスタの初期値は１で、この場合は通常のサンプリング動作を行います。

ｏｕｔｐ（ＢＡＳＥ＋０ｘ０Ｄ，ｃｎｔ０）　//　マルチサンプリング回数値（最下位バイト）
ｏｕｔｐ（ＢＡＳＥ＋０ｘ０Ｄ，ｃｎｔ１）
ｏｕｔｐ（ＢＡＳＥ＋０ｘ０Ｄ，ｃｎｔ２）
ｏｕｔｐ（ＢＡＳＥ＋０ｘ０Ｄ，ｃｎｔ３）　//　マルチサンプリング回数値（最上位バイト）

マルチサンプリング回数指定データをカウンター（オフセット：ＢＡＳＥ＋０ＤＨ）に書き込みます。

操作としては、必ず４バイト連続して書き込む必要があります。計数は３２ビットカウンター一本で行われ、計数値が“０”に到達すると、一回のマルチサンプリング動作が終了し初期値がリロードされ再びダウンカウントが始まります。

３－24　マルチサンプリングクロック源の選択

ｏｕｔｐ（ＢＡＳＥ＋０ｘ０Ｅ，ｍｃｓ）　//　マルチサンプリングクロック源を指定

表２４【ＢＡＳＥ＋０ｘ０Ｅ】出力ポートの構成

ビット	各ビットの機能・意味	＝１の時	＝０の時	リセット時
Ｂ７	外部クロック源の有効極性指定	↑（＋）	↓（－）	０
Ｂ６～Ｂ５	未使用			
Ｂ４	クロック源選択（外部／内部１０MHｚ）	外部（注１）	内部	０
B3　～B1	未使用			
B0	平均化処理実行の有無（注２）	実行する	実行しない	０

注１　マルチサンプリングクロック源の外部クロックとは、汎用デジタル入力のことで、通常のサンプリングクロック用の外部クロック端子とは異なります。

注２　“実行しない“を選択した場合、複数回分取り込まれた生データはそのままメモリに書き込まれ平均化等の処理はソフト処理によって行われることになります。

”実行する“を選択した場合、マルチサンプリングにより取り込まれたデータはハードウェアにより平均化され１データとしてメモリに書き込まれます。また、この平均化処理が可能なマルチサンプリング回数は２５５回までとなっています。

３－25　マルチサンプリングクロック源分周比指定

前節で説明したクロック源の分周比データを（ＢＡＳＥ＋０ｘ０Ｆ）レジスタに書き込みます。

必ず４バイト連続して書き込みを行う必要があります。分周は３２ビットのカウンターにより行われます。

ｏｕｔｐ（ＢＡＳＥ＋０ｘ０Ｆ，ｄｉｖ０）　//　分周比最下位バイト

ｏｕｔｐ（ＢＡＳＥ＋０ｘ０Ｆ，ｄｉｖ１）

ｏｕｔｐ（ＢＡＳＥ＋０ｘ０Ｆ，ｄｉｖ２）

ｏｕｔｐ（ＢＡＳＥ＋０ｘ０Ｆ，ｄｉｖ３）　//　分周比最上位バイト

第４章　ソフトウェア

４－１　本製品用ソフトウェアの種類

本製品用のソフトウェアは、表４－１に示す３種類から構成されています。即ち、WindowsXP(32Bit)対応のデバイスドライバー、Windows7(32Bit)対応のデバイスドライバー及びWindows7(64Bit)対応のデバイスドライバーの３種類です。また、夫々のＯＳ用に専用ハンドラー並びに汎用ＩＯハンドラーが用意されています。

表４－１　ドライバー構成表

ＯＳ及びビット幅		Windows7 32Bit	Windows7 64Bit	WindowsXP 32Bit
パッケージ名称		Redist516_x86.zip	Redist516_x64.zip	Redist516_xp.zip
専用ドライバー	ドライバー(DLL/Lib)	FCAD516PCI_X86	FCAD516PCI_X64	FCAD516PCI_X86
	関数宣言	FCAD516PCI.H	FCAD516PCI.H	FCAD516PCI.H
汎用ドライバー	ドライバー(DLL/Lib)	PORT7_X86	PORT7_X64	PORT7_X86
	関数宣言	PORT7.H	PORT7.H	PORT7.H

また、WindowsXP(32Bit)対応のドライバーは、アプリケーション移植時の利便性を考慮し用意されているものです。このドライバーを経由する事により、ドライバー更新→ＯＳ変更というステップを２段階に分割して実行できるように考慮されています。ドライバーそのものはWindows7(32Bit)版と同じものになっていますが、インストール方法の違いによりパッケージの構成がWindowsXP対応の形になっています。
またドライバー内の関数セット名称の内、同一機能のものについてはマイクロサイエンスのドライバーと同一とし移植時の利便性を考慮しています。

表４－１ａ　ドライバー構成表Ⅱ

ＯＳ及びビット幅		Windows7 32Bit	WindowsXP 32Bit
パッケージ名称		Redist516_X86_686.zip	Redist516_xp_686.zip
専用ドライバー	ドライバー(DLL/Lib)	H686z_2K/H686_2K	同左
	関数宣言	H686z.H/H686.H	同左
汎用ドライバー	ドライバー(DLL/Lib)	PORT_2K	同左
	関数宣言	PORT_2K.H	同左

表４－１ａで示される各ドライバーは、何らかの理由で、アプリケーション側の変更が困難な場合に使用して頂けるよう作成した製品です。ハンドラー内部コードについては、表４－１で示されるドライバーパッケージと同一ですが、Dll名称及びLib名称等、アプリケーションサイドから参照される情報についてもマイクロサイエンス製ドライバーと互換性を持たせた製品です。

2015年10月21日　マイクロサイエンス製ドライバー互換ドライバーの説明追加

４－２　ドライバーのインストール

４－２－１　WindowsXP ドライバーインストール及びアンインストール

WindowsXP 対応ドライバーをダウンロードし、その内容を全て解凍します。解凍場所は C ドライブのルートを指定して解凍します。解凍結果は以下のようになります。

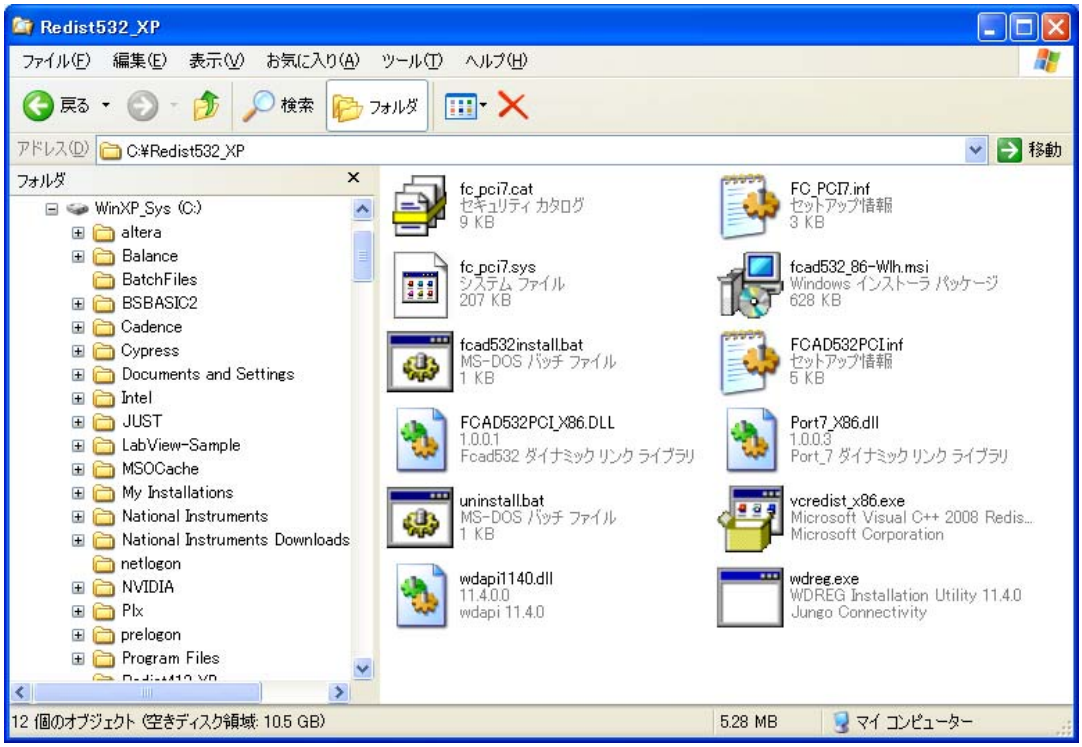


図４－２－１Ａ

デバイスインストールを開始する前に、旧ドライバーがインストールされていた場合、これを削除しなければなりません。そのためには、デバイスマネージャを開き、マイクロサイエンスドライバーが登録されているかどうかを確認します。因みに、下の図は、参考例としてＨＤＡ－７７０ＰＣＩがインストールされている状態で説明しています。

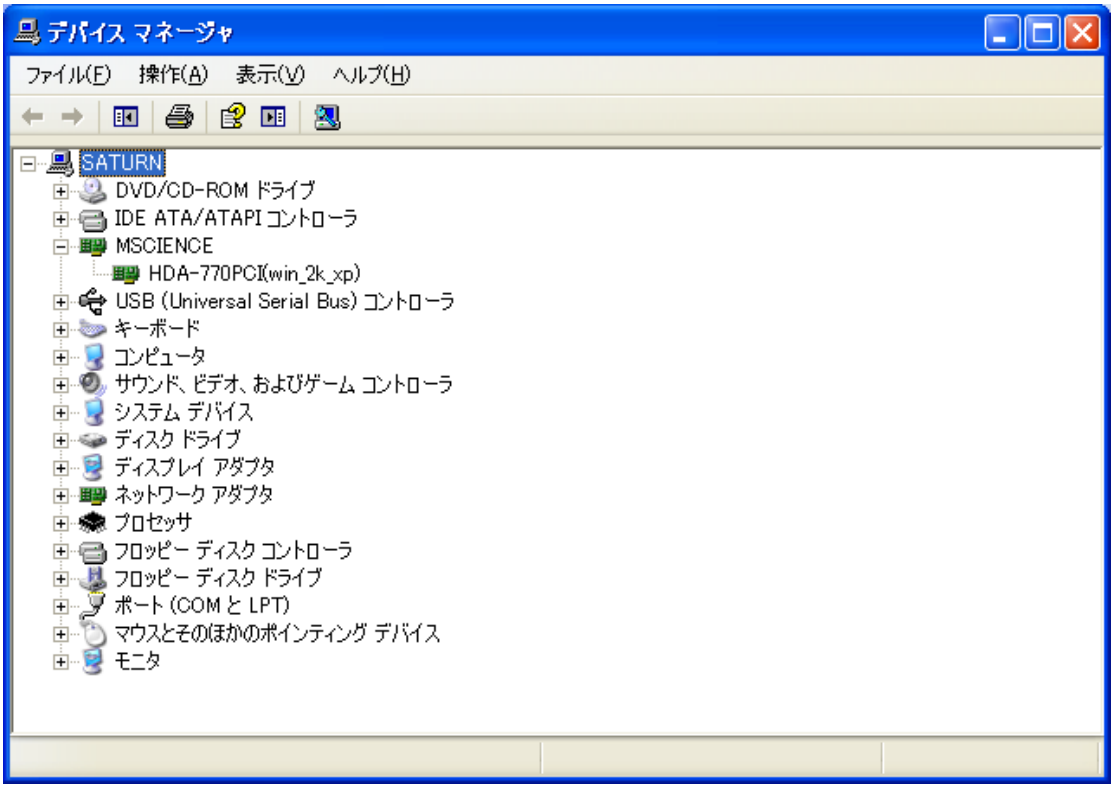


図４－２－１Ｂ

上図に示すようにMSCIENCEというエントリーが存在すれば、旧ドライバーがインストールされています。
ドライバー削除は、

1　まずWindows¥Infフォルダの中からOem*.Infファイルを検索します。検索条件としては、文字列として“MSCIENCE”が含まれているものを条件とします。検索されたファイルをテキストエディターで開きマイクロサイエンスのものである事を確認した後、Oemxx.Inf（xx は上記の条件に合致した番号です）とOemxx.Pnfを削除します。
2　次にWindows¥System32フォルダの中から、DMS_PCI.Sysを探し同じように削除します。
3　最後に、図４－２－１Ｂで示すマイクロサイエンス製ドライバー（例ではHDA-770PCIです）を右クリックし削除します。

ここまできた段階で一旦パソコンの電源を落とし、ボードを取り外します。
再度パソコンの電源を投入し、図４－２－１Ａで示されるフォルダの中から“Fcad516PciInstall.Bat”ファイルを右クリックし“開く”（実行する）を選択します。すると、まず図４－２－１Ｃで示されるウィンドーが開きます。（次ページ）

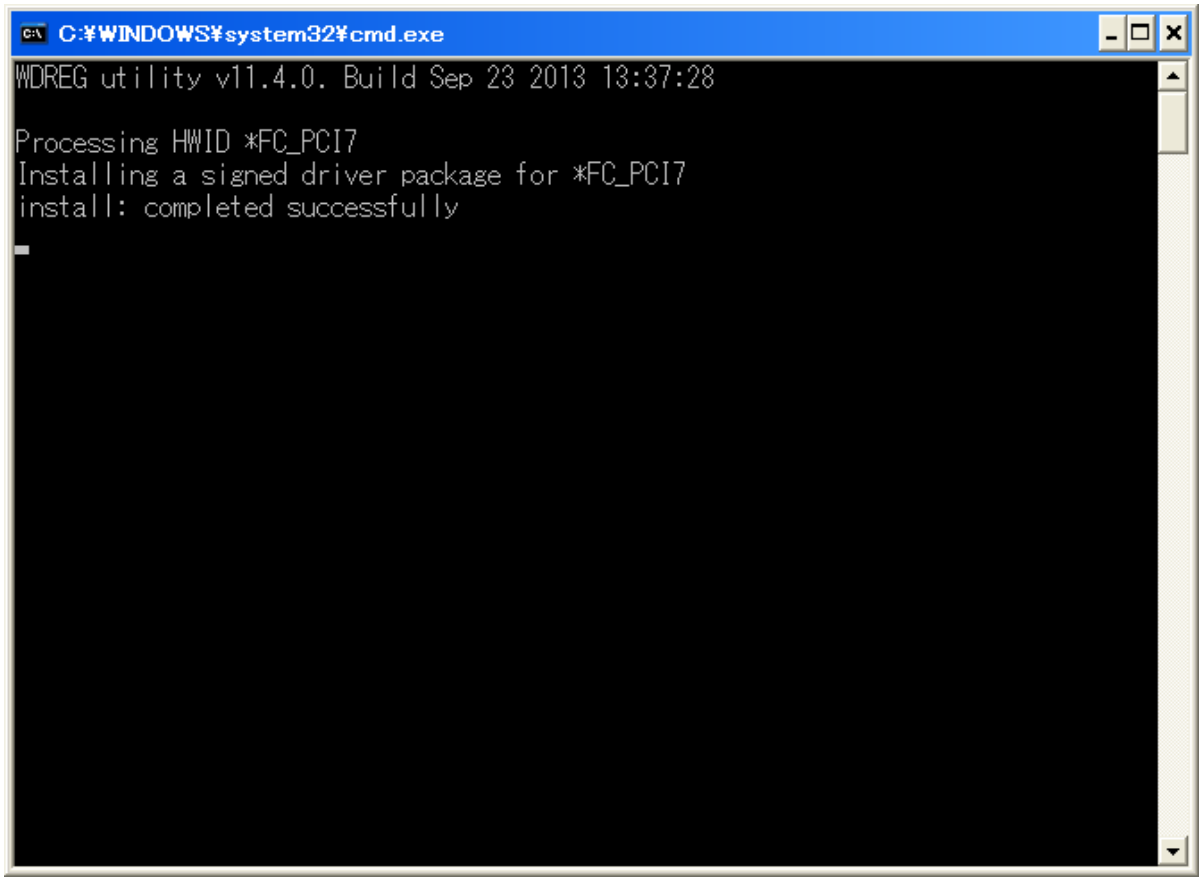
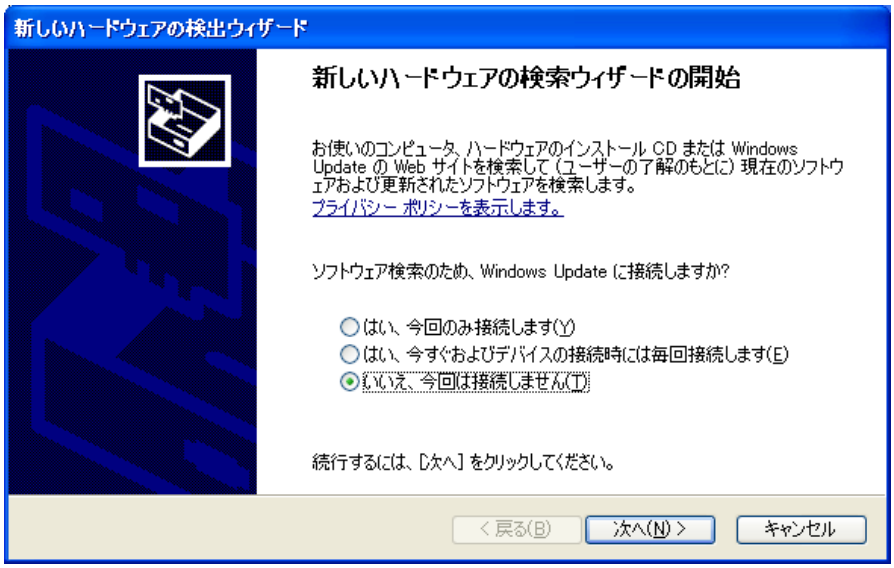
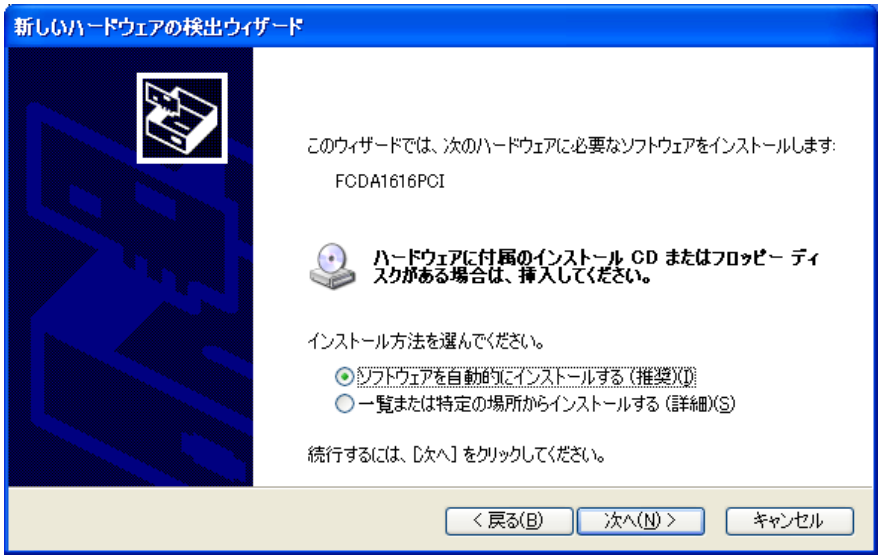


図４－２－１C

そして暫くすると、インストール作業が終了します。そこで再度パソコンの電源を遮断し FCAD516PCI ボードを再度挿入し電源を投入します。システムが立ち上がるとハードウェアウィザードが立ち上がりますから、



WindowsUpdate への接続をせず、“次へ“ボタンをクリックします。そして次の画面（次頁）で、次のハードウェアとして” FCAD516PCI” が表示された場合は、インストール方法として“ソフトウェアを自動的にインストールする”を選択し“次へ“ボタンをクリックします。すると必要な Dll ファイル群がシステムフォルダにコピーされインストール作業は終了します。また次頁のようにハードウェアとして”DPIO モジュール”が表示された場合は”一覧または特定の場所からインストールする“を選択し”次へ“ボタン押下後の画面でインストール先として“C:¥Redist516_XP”を選択しインストールを行います。



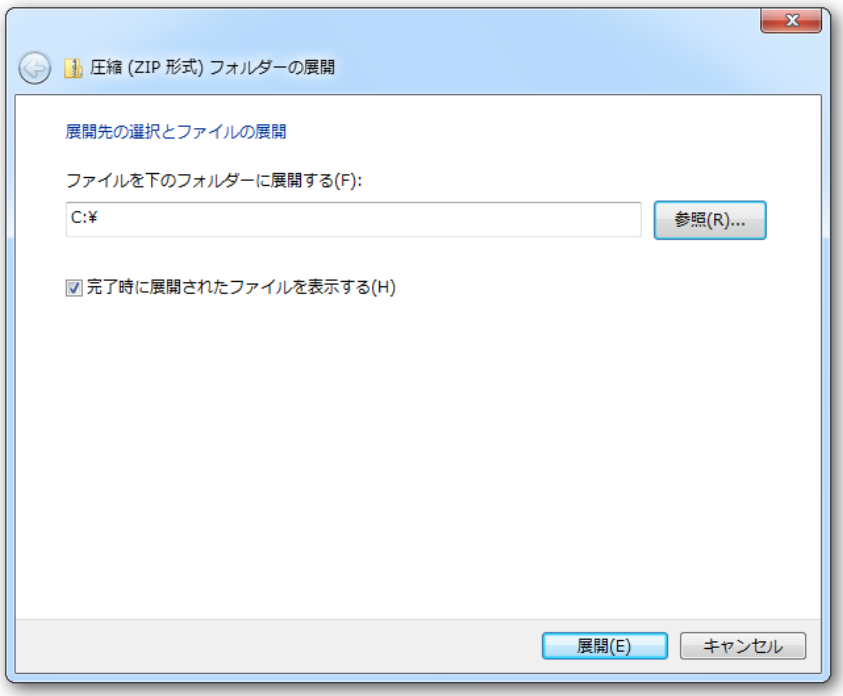
推奨設定のまま、“次へ”ボタンをクリックします。すると必要なDllファイル群がシステムフォルダにコピーされインストール作業は終了します。
ドライバーを削除する場合は、図４－２－１Ａで“Uninstall.Bat”ファイルを実行することで自動的にドライバーが削除されます。

４－２－２　Windows７（X86：32ビット）へのインストール

本製品用のソフトウェアは弊社ホームページ内ダウンロードコーナーにアップロードされています。その中から、実際に御使用になるＯＳとマッチしたドライバーセットをダウンロードして頂き本ボードを取り付ける前にデバイスドライバーをインストールしていただきます。

実際のインストール方法は以下の通りです。

・　本ボードをパソコンにセットアップする前に、弊社ホームページよりご使用のＯＳ用デバイスドライバーキットをダウンロードして頂きパソコンの中に解凍します。解凍場所は、Ｃドライブのルートとします。（本ドライバーキットには、デバイスハンドラーが含まれています。）



前ページの“展開”ボタンをクリックするとドライバーセットがＣドライブ内に展開されます。内容は図４－２－２Ａを参照してください。

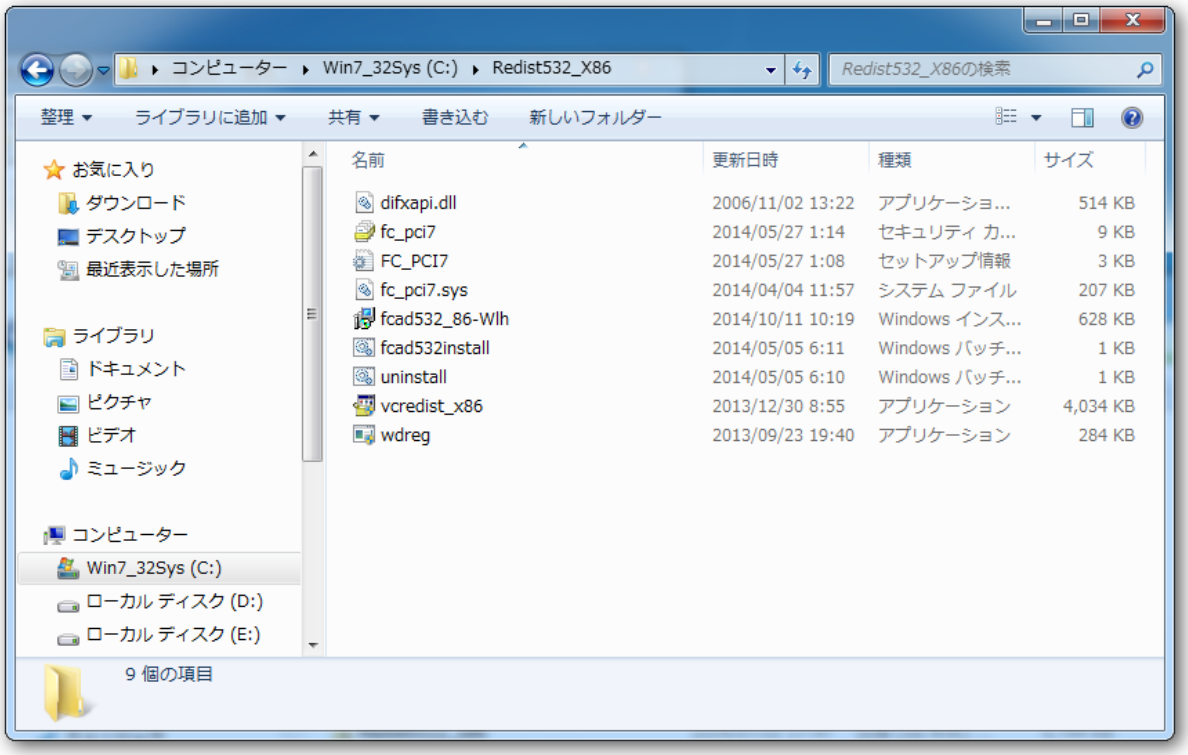
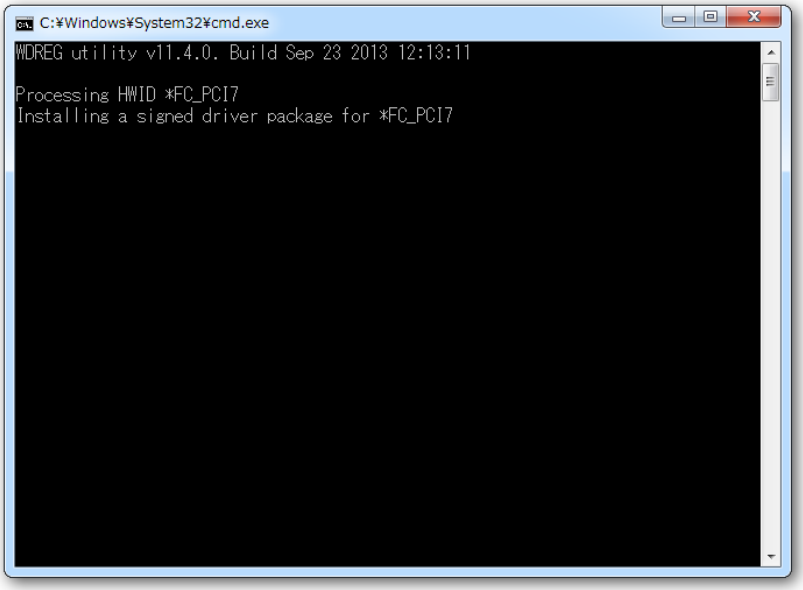


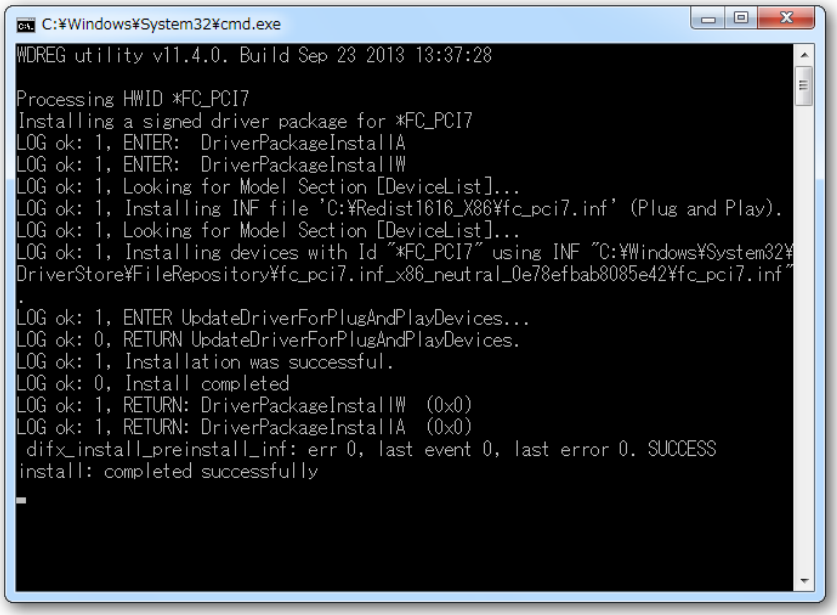
図４－２－２Ａ

この中で“Fcad516Install.Bat”がデバイスドライバーインストーラーです。***また、この時点で、本ボードは、まだパソコンには取り付けないで***下さい。この順序が守られない場合、正常なインストールは保証されません。このデバイスドライバーインストーラーを***管理者として***実行（右クリックメニューから選択）すると次のウィンドーが開きます。（ただ実行とすると、ウィンドーが瞬間的に終了しデバイスドライバーのインストールはできません）

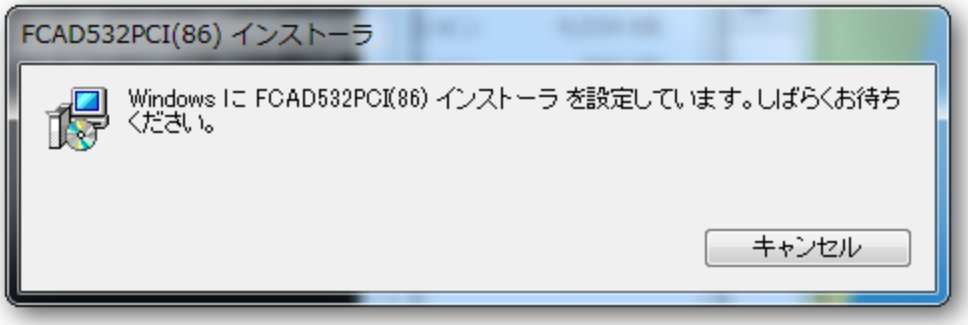
最初に次のウィンドーが開きます。



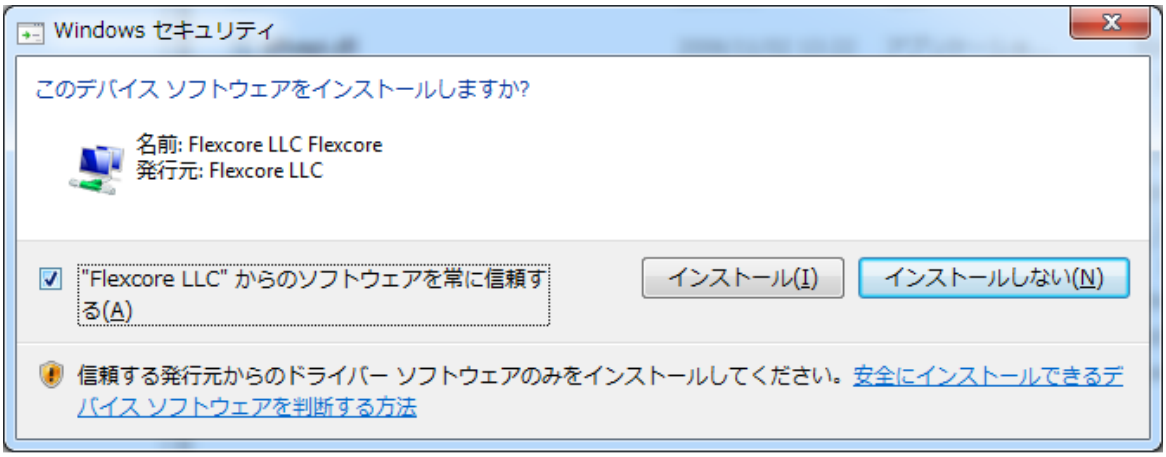
次いで次ページに示すウィンドーが開きます。



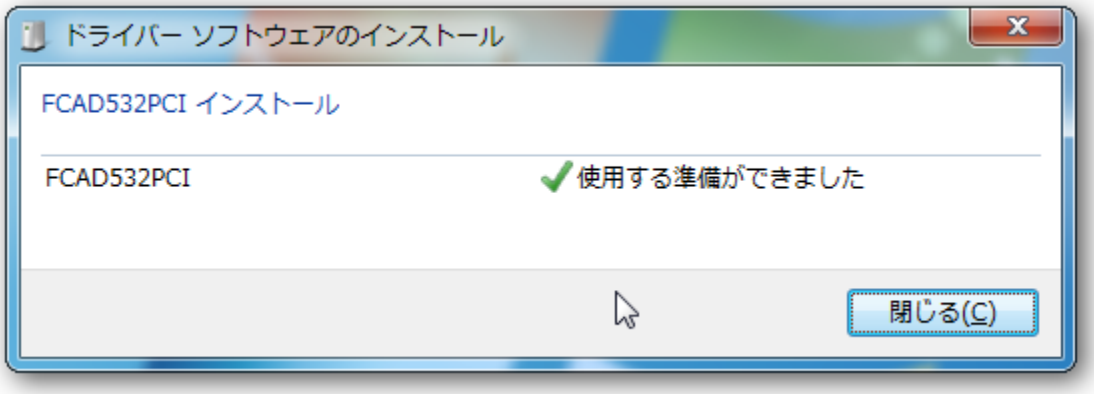
更に暫く待っていると、次のウィンドーが開きます。



更に待っていると次のウィンドーが開きます。



ここでは、“Ｆｌｅｘｃｏｒｅ　ＬＬＣ”からのソフトウェアを常に信頼するという項目にチェックを入れ“インストール”ボタンをクリックします。そして暫く待っているとインストール作業が終了します。
この後、一旦パソコンの電源を落とし、パソコン内部のＰＣＩスロット或いはＰＣＩ拡張ユニットに本ボードを挿入し再度パソコンの電源を投入します。ＯＳが立ち上がるとハードウェアの検出が行われ、少し待っていると画面下部のタスクバーの所に、デバイスドライバーが正常にインストールされたというメッセージが表示されます。



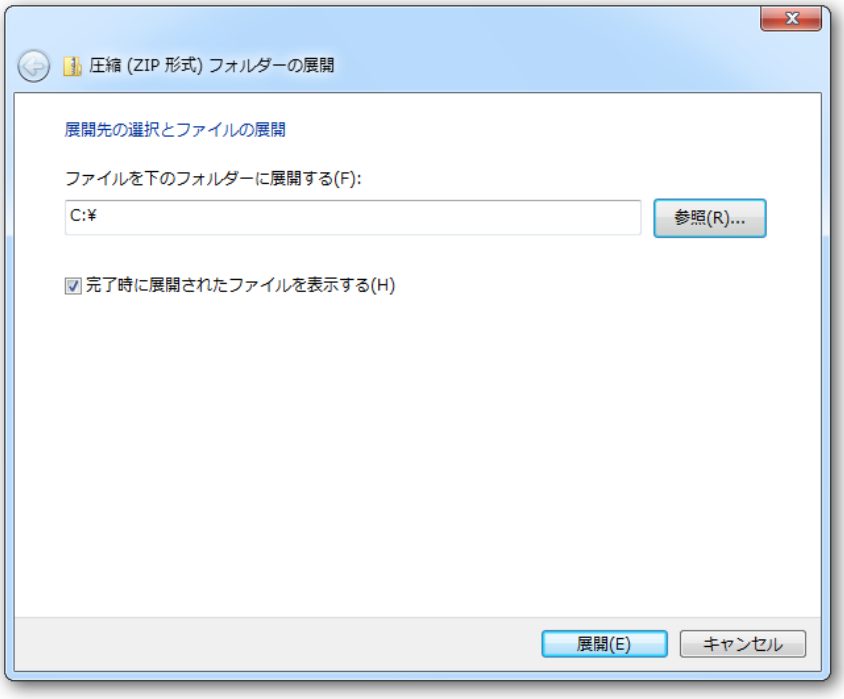
これで本ユニットのインストールは完了しました。

４－２－３　Windows７（X64：64 ビット）へのインストール

本製品用のソフトウェアは弊社ホームページ内ダウンロードコーナーにアップロードされています。その中から、実際に御使用になるＯＳとマッチしたドライバーセットをダウンロードして頂き本ボードを取り付ける前にデバイスドライバーをインストールしていただきます。

実際のインストール方法は以下の通りです。

・　本ボードをパソコンにセットアップする前に、弊社ホームページよりご使用のＯＳ用デバイスドライバーキットをダウンロードして頂きパソコンの中に解凍します。解凍場所は、Ｃドライブのルートとします。（本ドライバーキットには、デバイスハンドラーが含まれています。）



ここで“展開”ボタンをクリックするとドライバーセットがＣドライブ内に展開されます。内容は図４－２－３Ａを参照してください。

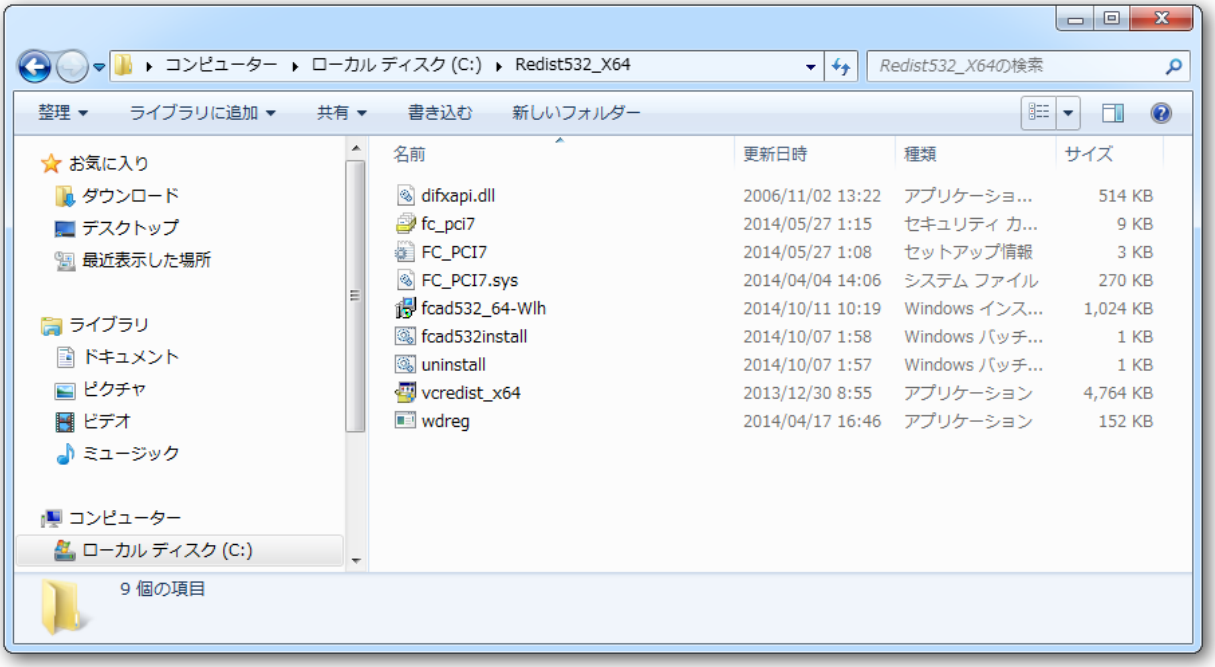
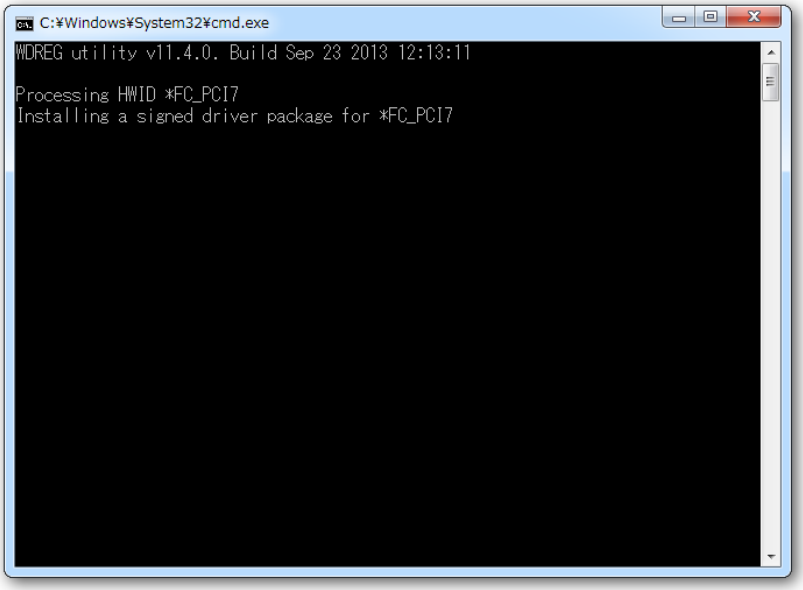


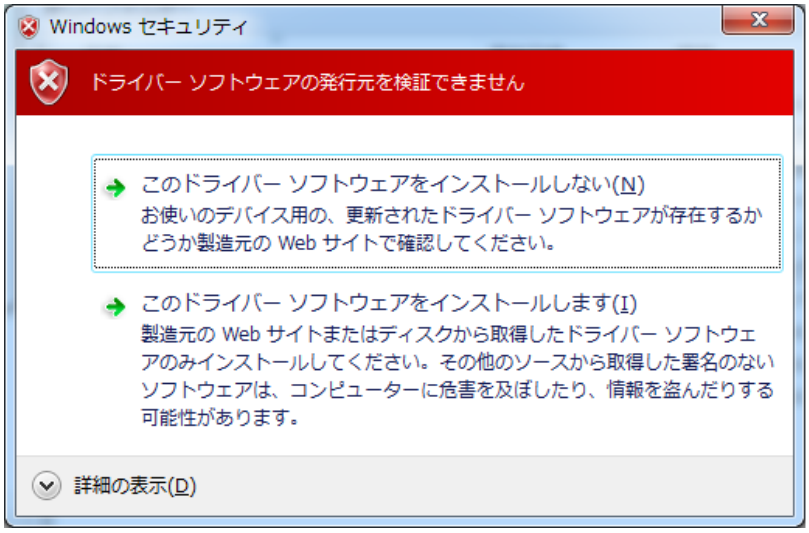
図４－２－３Ａ

この中で“Fcad516Install.Bat”がデバイスドライバーインストーラーです。***また、この時点で、本ボードは、まだパソコンには取り付けないで***下さい。この順序が守られない場合、正常なインストールは保証されません。このデバイスドライバーインストーラーを***管理者として***実行（右クリックメニューから選択）すると次のウィンドーが開きます。（ただ実行とすると、ウィンドーが瞬間的に終了しデバイスドライバーのインストールはできません）

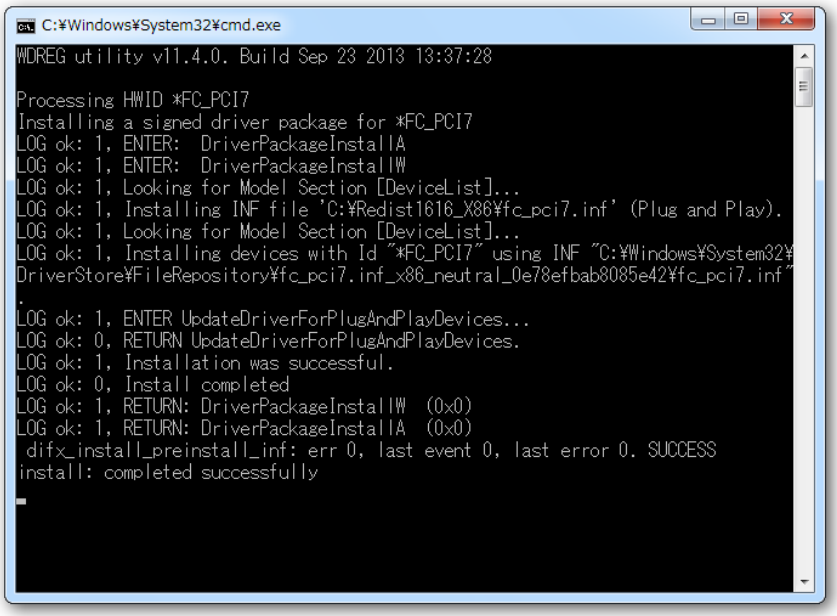
最初に次のウィンドーが開きます。



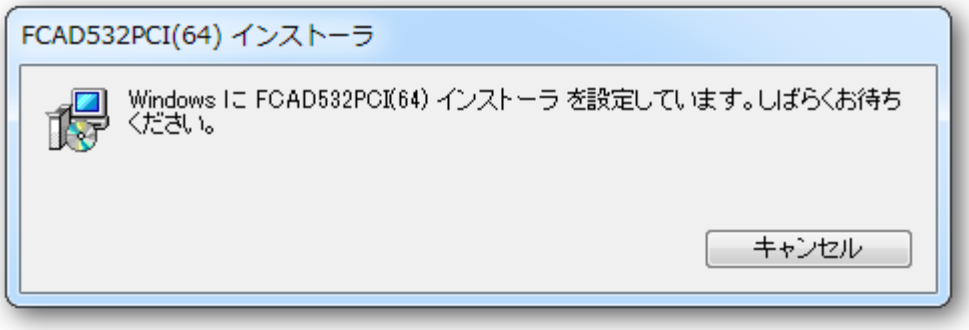
次いで次ページに示すウィンドーが開きます。



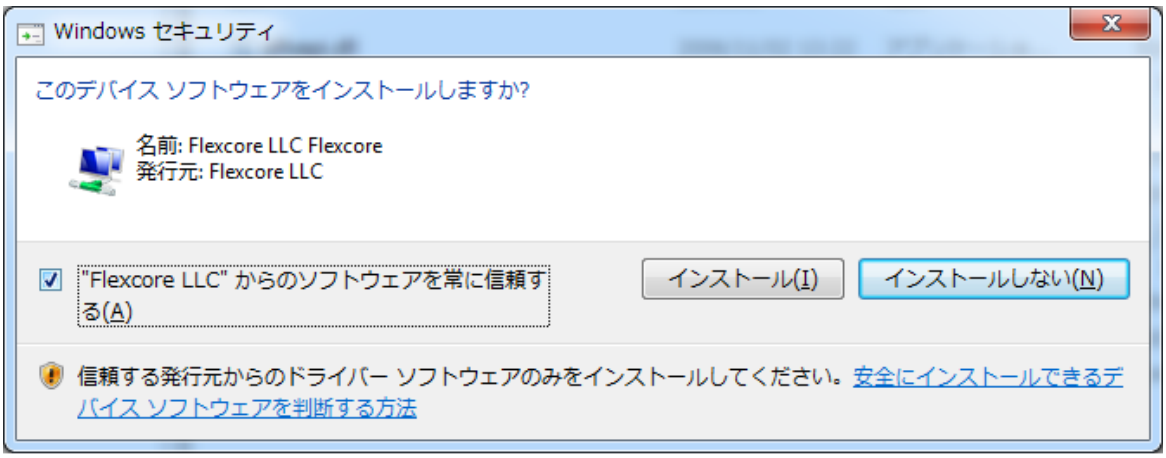
ここでは、“このドライバーソフトウェアをインストールします”側のボタンをクリックして下さい。すると次のウィンドーが表示されます。



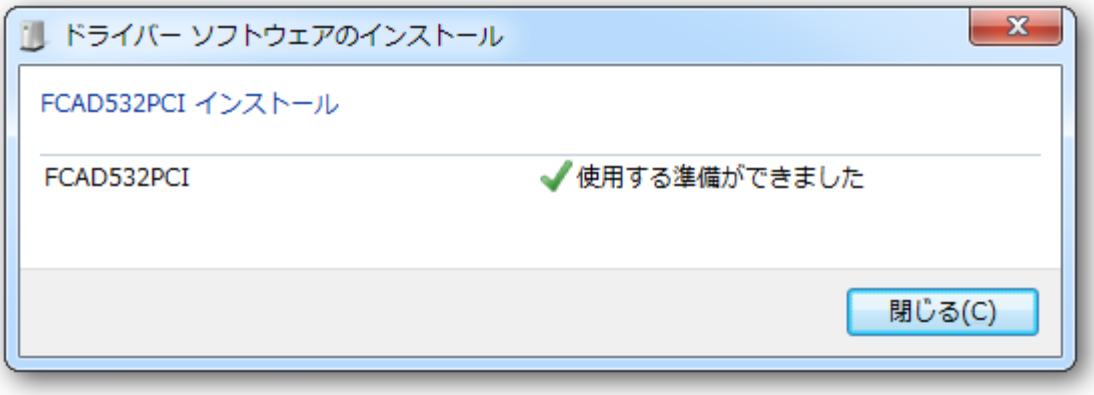
更に暫く待っていると、次のウィンドーが開きます。



更に待っていると次のウィンドーが開きます。



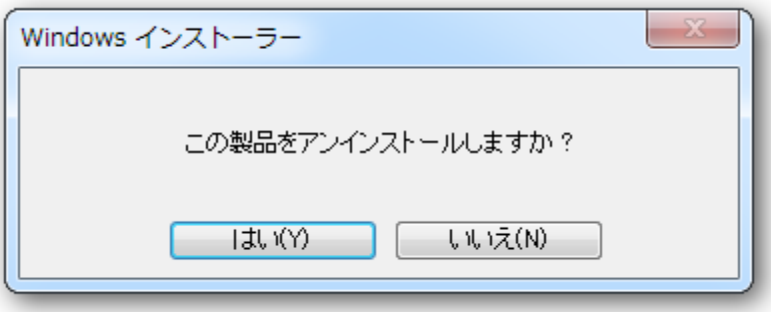
ここでは、“Ｆｌｅｘｃｏｒｅ　ＬＬＣ”からのソフトウェアを常に信頼するという項目にチェックを入れ“インストール”ボタンをクリックします。そして暫く待っているとインストール作業が終了します。
この後、一旦パソコンの電源を落とし、パソコン内部のＰＣＩスロット或いはＰＣＩ拡張ユニットに本ボードを挿入し再度パソコンの電源を投入します。ＯＳが立ち上がるとハードウェアの検出が行われ、少し待っていると画面下部のタスクバーの所に、デバイスドライバーが正常にインストールされたというメッセージが表示されます。



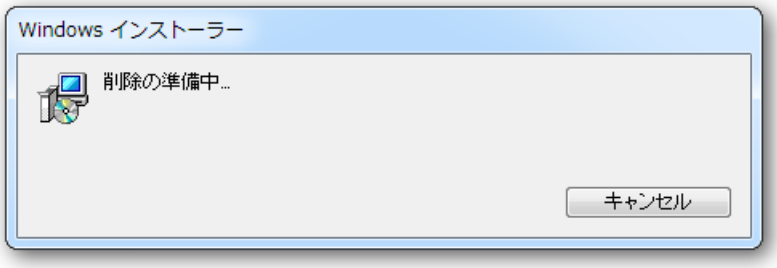
これで本ユニットのインストールは完了しました。

４－３　Windows７ドライバーのアンインストール

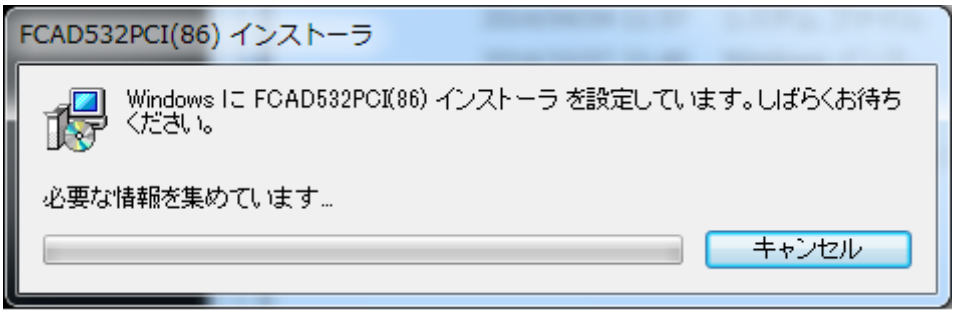
この方式でインストールしたドライバーはドライバー更新時などにドライバーをアンインストールする際にも簡単に行う事ができます。まず、４－２－１項で説明したデバイスドライバー解凍フォルダへ移動し、この中にある“Uninstall.Bat”ファイルを実行します。（ここでも*管理者として*実行する必要があります。）



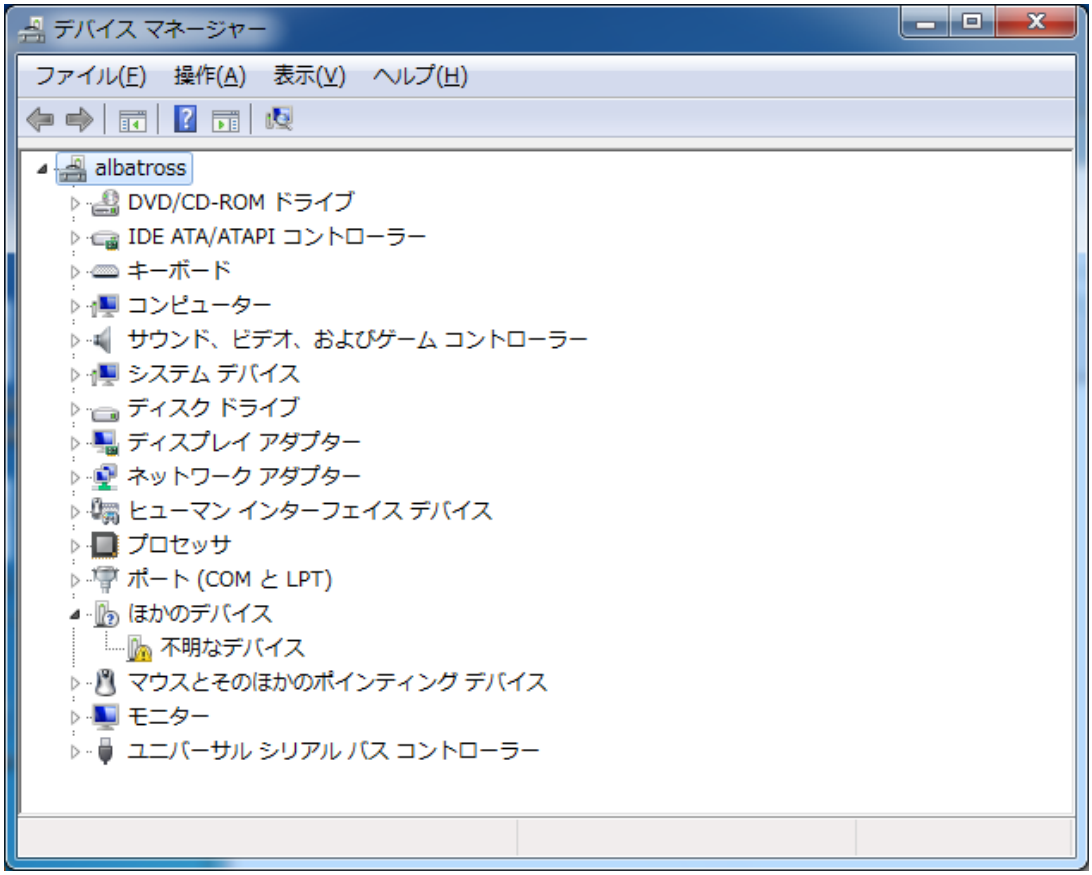
というメッセージが現れますので“はい”をクリックします。すると



というウィンドーに変わり、更に待っていると

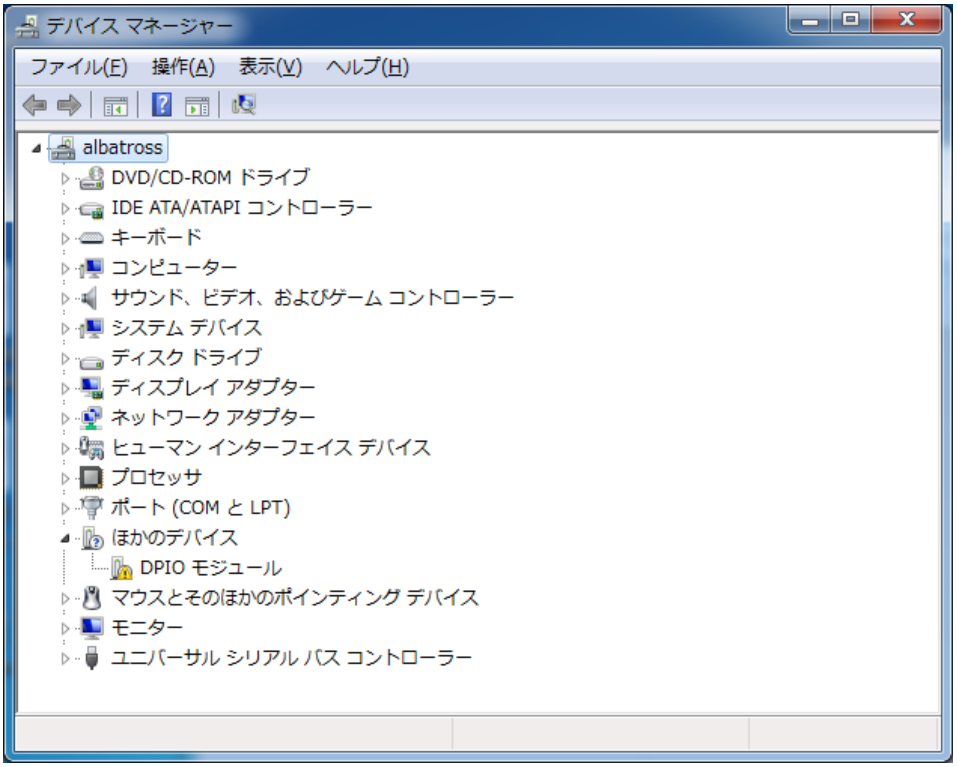


というウィンドーに変わり、暫く待っているとデバイスドライバーが削除されます。
この段階で、スタートー＞コントロールパネルと進み、デバイスマネージャを開くと、下図に示すように不明なデバイスが表示されています。（同図下から４行目）



そこで、この不明なデバイスを右クリックし、表示されるドロップダウンリストから“デバイスの削除”を選び実行します。その結果、次ページに示すように不明なデバイスが、ドライバー未インストール時と同様、ほかのデバイスの中にＤＰＩＯモジュールが存在する状態に変化します。（この表示を得るためには、トップエントリ：同図では albatross と表示されているエントリーを右クリックし“ハードウェア変更のスキャン”を実行する必

要があります)
この状態に到達すると、パソコンの電源を遮断し、本ボードを取り外す事ができます。
（基本的には、前頁で示したハードウェア変更のスキャンを実施することなく、パソコンの電源を遮断し、当該ボードを取り外す事ができますが、念のためハードウェア変更のスキャンを実行されることを推奨します。）

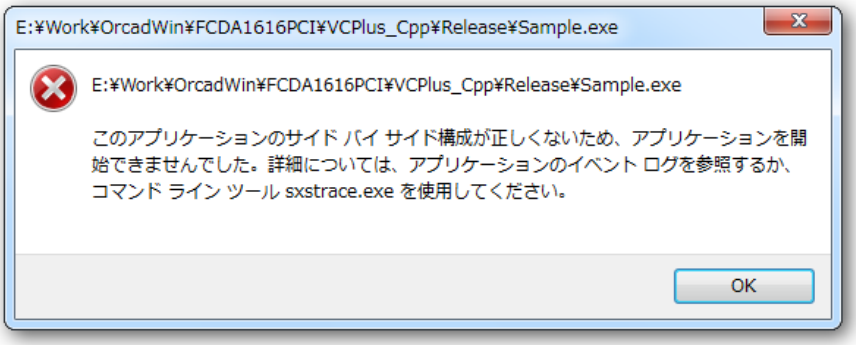


４－４　サンプルプログラムについて

本製品のサンプルプログラムは、大きく２種類から構成されています。第一のプログラムは、汎用入出力関数を利用したボードの動作確認プログラムであり、(内部レジスタ動作確認及び簡単な AD データ取り込み処理)第二のプログラムは専用ハンドラーを使用した C+による自動サンプリング（Dll バッファー使用)、マニュアルサンプリング（ユーザーバッファー使用）及びループ処理プログラムの３構成から構成されています。また、アナログ回路設定変更などの実行例については､基本的に自動サンプリングサンプルの中で記述しています｡また、X86（32 ビット）及び X64(64 ビット)の実行ファイルについてはそれぞれのサンプルプログラム内でディレクトリ構造によって住み分けを行っています。詳細については、弊社ホームページよりダウンロードし内容の確認をお願いします。

４－５　サンプルプログラム、動作確認プログラムについて（補足）

サンプルプログラムを実際に動作させると､以下のようなメッセージが出て動作しない事があります｡これは､Windows７若しくはWindowsXPにおいてＣ＋ランタイムが標準でインストールされていないためです。



Windows7の場合

WindowsXPの場合

弊社で作成した動作確認プログラムはＭＳＶＣ＋２００８でコンパイルされておりますので、専用のランタイムをインストールする事により、この状況を回避する事ができます。必要となるランタイムは、ドライバーパッケージの中に同梱しておりますので、こちらを実行することでランタイムのインストールができます。ファイル名称は、ｖｃｒｅｄｉｓｔ_ｘ８６.ｅｘｅ（Ｘ８６環境）又はｖｃｒｅｄｉｓｔ_ｘ６４.ｅｘｅ（Ｘ６４環境）になります。

若しくは、（まだＣ＋環境をインストールされていない場合）当該パソコンにＭＳＶＣ＋２００８をインストールする事でも対処する事ができます。

４－６　関数仕様

本ボードの関数セットは、表４－４に示すように、マイクロサイエンス製ドライバー内の関数名と等価な関数及び名称が全く異なる関数群から構成されています。これは、アプリケーション移植に際して意味のない名称修正等を避け、移植ミスをできるだけ減らすためです。名称に互換性がない関数は、本ボードでハードウェア上追加された機能を使用する関数若しくは元々の関数セットには存在しなかった関数に限られていますので、アプリケーション上これらの機能を使用する必要がない場合、アプリケーション側の修正箇所は、ハンドラーDll 名称、インポートライブラリ名称の変更、インクルードファイルの変更及びメッセージ ID 番号の変更のみでドライバー移植を完成する事ができます。また、本ボードで追加されたハードウェア機能を採用される場合は、同表後半の明らかに異なる関数名称を付与された関数群を混在させて追加採用する事で実現が可能です。

表４－４　関数仕様

連番	関数名	機能・動作	パラメータ等
【１】	AD686z_Open_AdSys	ボード、本ハンドラーの初期化	使用ボード数
【２】	AD686z_Set_SampCh	サンプリング実行チャンネル数設定	実行チャンネル数
【３】	AD686z_Set_SampMode	サンプリングモードの設定	バッファー領域
【４】	AD686z_Set_Trigger	トリガ関連の設定	トリガ源、極性
【５】	AD686z_Set_RangeTrigger	レンジトリガ条件設定	
【６】	AD686z_Set_MultiScan	マルチサンプリング設定	
【７】	AD686z_Set_ExClk	オプション外部クロック値の設定	外部クロック周波数
【８】	AD686z_Set_Clock	クロック源、クロック値の設定	クロック源、分周比
【９】	AD686z_Set_Mexclk	マルチサンプリング外部クロック値設定	
【10】	AD686z_Set_MsClock	マルチサンプリングクロック設定	
【11】	AD686z_Start_Samp	サンプリングスタート	
【12】	AD686z_Close_AdSys	本ハンドラーの終了	
【13】	AD686z_Get_Status	ボードステータスの取得	ボードステータス
【14】	AD686z_Read_DllData	DLL バッファーからのデータ読出し	波形データ入力
【15】	AD686z_Read_DirectFifo	ユーザーメモリへの直接データ転送	波形データ入力
【16】	AD686z_Get_OneScan	マニュアルサンプリング	ＡＤサンプリングデータ
【17】	AD686z_Out_Aux	汎用デジタル出力	更新データ
【18】	AD686z_Inp_Aux	汎用デジタル入力	
【19】	AD686z_Set_SampLoop	リングバッファ指定有無	
【20】	AD686z_Set_Inpmode	入力モード設定	
【21】	AD686z_Stop_Samp	サンプリングの強制停止	
【22】	AD686z_Clear_Flags	ボードステータスのクリア	
【23】	AD686z_Read_RestData	サンプリング残りデータの取得	
【24】	AD686z_Set_Shc	サンプルホールド回路パラメータセット	
【25】	AD686z_Set_ScanSpeed	スキャン速度設定	
【26】	AD686z_Plus_Message	追加メッセージ指定	
【27】	AD686z_Get_Libver	本ハンドラーのバージョン情報取得	
	以下は本ボード専用関数（マイクロサイエンスボードでは実装されていないハードウェアに関する）の説明である		
【28】	AD516_Get_HalfSize	ハーフフルサイズを取得する関数	
【29】	AD516_Get_UsedSize	メモリ使用データ数の取得	
【30】	AD516_Set_DataCode	AD データコード、入力モード等の設定	入力レンジ、レンジモード
【31】	AD516_Get_DataCode	AD データコード、入力モード等の取得	入力レンジ、レンジモード

【1】ボード及び本ハンドラーの初期化

int __stdcall AD686z_Open_AdSys (HWND Owner, int Num_Board, int *fifo_size);	
引数	Owner:ウィンドーハンドル Num_Board:使用ボード枚数 fifo_size:FIFO メモリのハーフフルサイズを報告するためのポインター
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	プラグアンドプレイで設定された各ボードのＩ／Ｏアドレス、及び割り込みレベルを認識すると同時にボードリセット、その他ハンドラー内の各種テーブルやデータバッファ等を初期化する。また、複数ボードが存在する場合枚数及び夫々のボード番号をアプリケーション側へ報告する。

【2】サンプリングチャンネル数設定

int __stdcall AD686z_Set_SampCh(int board_no, int no_ch, int scan_ordr[], int range[]);	
引数	board_no:ロータリースイッチの値（０から始まる連続数値） no_ch:サンプリングチャンネル数 scan_ordr[]:未使用 range[]:未使用
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	ロータリースイッチ番号 board_no のボードに、サンプリングチャンネル数を設定する

【3】サンプリングモード指定

int __stdcall AD686z_Set_SampMode (int trs_trig, int buf_area, int intr_sw);	
引数	trs_trig:AD データ転送の参照フラグ、０：ノットエンプティ、１：ハーフフルサイズ buf_area:AD データ転送先、０：DLL バッファー、１：ユーザープログラム内バッファー intr_sw:割り込みイベント発生要因指定（現状ダミー）
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	trs_trig を０とするとＡＤデータ１個毎の転送、１とするとハーフフルサイズ毎の転送となる。 この時 buf_area を０と指定すると DLL バッファーへ転送が行われ、１とするとユーザープログラム内バッファーへユーザーが転送する必要がある。intr_sw については現状ダミー扱いである

【4】トリガ関連の設定

int __stdcall AD686z_Set_Trigger (int trg_mode, int trg_source, int trg_pol, int trg_level);	
引数	trg_mode:サンプリングモード（０：即トリガ、１：ポストトリガ） trg_source:トリガーソース指定（０：ソフトトリガ、１：アナログトリガ、２：デジタルトリガ） trg_pol:トリガ極性（０：負エッジ、１：正エッジ、２：負レベル、３：正レベル） trg_level:トリガレベル（対応するアナログ値上位８ビットで指定する）
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	ロータリースイッチ番号０のボードに、トリガ条件を設定する。またスレーブボードに対してはトリガーソースとタイプ指定について固定の条件が設定される。（ソフトトリガ、立ち下がりエッジ）それ以外の条件は指定されたものを使用する。

【５】レンジトリガ設定

int __stdcall AD686z_Set_RangeTrigger (int trg_sel, int trg_level_hi, int trg_level_lo);	
引数	trg_sel:アナログトリガモード、０：アウトレンジ、１：インレンジ、２：デュアルスロープ（＋） ３：デュアルスロープ（－） trg_level_hi:トリガレベル上限値 trg_level_lo:トリガレベル下限値
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	レンジトリガ関連の設定を行う。

【６】マルチサンプリング動作モード設定

int __stdcall AD686z_Set_MultiScan (int no_msamp, int do_average);	
引数	no_msamp:マルチスキャン回数、１～$2^{32}-1$ do_average:平均化処理実行の有無、０：平均化しない、１：平均化する
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	全てのボードに対してマルチサンプリング動作モードを設定する

【７】サンプリングクロックオプション或いは外部クロック源周波数の設定

int __stdcall AD686z_Set_Exclk(long exclk_freq);	
引数	exclk_freq:オプションまたは外部クロックの周波数値（単位：Hz）
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	オプションクロックまたは外部クロック源を使用するとき、その周波数値を設定する。分周比によるクロック設定または標準クロック使用の場合本関数を操作する必要は無い。

【８】クロック源及びクロック値の設定

int __stdcall AD686z_Set_Clock (int clk_source, int set_mode, int *time_unit, int *clk_period);	
引数	clk_source:クロック源　０：内部クロック０（１０MHｚ） １：内部クロック１（８．１９２MHｚ） ２：内部クロック２（オプション）、３：内部クロック３（未使用） ４：外部クロック立下りエッジ、５：外部クロック立上りエッジ set_mode:クロック値の設定方法　０：周期の値、１：分周比 time_unit:クロック周期の単位　０：秒、１：m秒、２：u秒、３：n秒 clk_period:クロック周期の値、または分周比
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	ロータリースイッチ番号０のボードに、クロック設定条件を通信する。またスレーブボードには固定の条件を設定する。（外部クロック立ち下がりエッジ、分周比設定、分周比＝１）

【９】マルチサンプリング用外部クロック値の設定

int __stdcall AD686z_Set_Mexclk (int mexclk_freq);	
引数	mexclk_freq:マルチサンプリング用外部クロック源の周波数値（単位：Hz）
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	マルチサンプリング実行タイミングに外部クロック源（汎用デジタル入力 D0）を使用するとき、その周波数値を設定する。分周比によるサンプリングクロック設定若しくは標準クロックによるサンプリングの場合本関数を操作する必要はない

【10】マルチサンプリングクロック値の設定

int __stdcall AD686z_Set_MsClock (int clk_source, int set_mode, int *time_unit, int *clk_period);	
引数	clk_source:マルチサンプリングクロック源選択 ０：内部クロック源０（１０MHｚ） ４：外部クロック源立下りエッジ、５：外部クロック源立上りエッジ set_mode:クロック源の指定方法　０：クロック周期の値、１：分周比 time_unit:クロック周期の単位　０：秒、１：m秒、２：u 秒、３：n秒 clk_period:クロック周期の値または分周比
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	マルチサンプリングクロック値を設定する。クロック源として外部クロック（汎用デジタル入力）をクロック周期で使用する場合、AD687z_Set_Mexclk 関数によりクロック源周波数値を定義しておく必要がある。

【11】サンプリング開始

int __stdcall AD686z_Start_Samp (int no_samp);	
引数	no_samp:チャンネルあたりのトリガ後サンプリング点数
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	関数【１】から【10】によって設定された条件で連続・自動サンプリングを行う。Dll バッファーを使用する際には、この値×チャンネル数、更に平均化しないマルチサンプリングを行う場合には更にその回数をかけた値だけのバッファー領域が確保される。

【12】ハンドラー終了

int __stdcall AD686z_Close_ADSys (void);	
引数	なし
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	本ハンドラーを終了し確保したメモリ領域の開放等を行う。

【13】ステータス取得

int __stdcall AD686z_Get_Status (int *sampled, int status[]);	
引数	sampled:サンプリング済みチャンネルあたりデータ点数 status[0]:ＡＤボードのステータス１ status[1]:ＡＤボードのステータス２
戻り値	正常終了時：０（連続サンプリング実行中)、１：停止中 エラー発生時：エラーコード（負の値)
機能・動作	本関数実行時のサンプリング済みデータ点数、及びマスターボードのステータス生データを得る。

【14】ＤＬＬバッファーからのＡＤデータ転送

int __stdcall AD686z_Read_DllData (int board_no, int no_data, int data_pos, UINT_PTR *bufptr, int bufsize);	
引数	board_no:対象ボード番号 no_data:Dll バッファーから読み出すチャンネルあたりのデータ点数 data_pos:Dll バッファーから読み出すデータの先頭位置 bufptr:転送先ＡＤデータバッファのポインター bufsize:転送先ＡＤデータバッファの大きさ（バイト指定)
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値)
機能・動作	サンプリング済みＡＤデータを Dll バッファーの任意の部分からユーザーバッファーへ転送する。

【15】ボードからユーザープログラム内バッファーへの直接転送

int __stdcall AD686z_Read_DirectFifo(int board_no, UINT_PTR *bufptr, int bufsize);	
引数	board_no:通信の相手となるボードの番号（SW-BN の値)
	bufptr:転送先ＡＤデータバッファのポインター
	bufsize:転送先ＡＤデータバッファのサイズ（単位：バイト＝総データ数×２）
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値)
機能・動作	ＡＤボードから直接ユーザープログラム内のバッファーへＡＤデータを転送する。転送タイミングは Get_Status 関数でハーフフル状態を検知するか、該当する条件で発行されるメッセージをトリガとして行う。将来的には、割り込みによる処理も検討している。

【16】マニュアルサンプリングスキャン

int __stdcall AD686z_Get_OneScan(UINT_PTR *bufptr, int bufsize);	
引数	bufptr:ＡＤデータバッファのポインター
	bufsize:ＡＤデータバッファのサイズ（単位：バイト＝総データ数×２）
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値)
機能・動作	システム内の全ボードに対して一回だけサンプリングスキャンを実行する。マルチサンプリング動作も加味する事は可能だが、マルチサンプリングクロック値は無効となり間断ない繰り返し動作となる。

【17】汎用デジタル出力

int __stdcall AD686z_Out_Aux(int board_no, int out_data);	
引数	board_no:通信の相手となるボードの番号（SW-BN の値） out_data:汎用１ビットデジタル出力データ
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	指定したボードから、汎用出力を更新出力する

【18】汎用デジタル入力

int __stdcall AD686z_Inp_Aux(int board_no);	
引数	board_no:通信の相手となるボードの番号（SW-BN の値）
戻り値	正常終了時：汎用１ビットデジタル入力データ エラー発生時：エラーコード（負の値）
機能・動作	指定したボードについて、汎用入力を取得する

【19】ＤＩＩバッファーのリングバッファ化制御

int __stdcall AD686z_Set_SampLoop(void);	
引数	なし
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	Ｄ１１バッファーのリング化オンオフトグルスイッチ機能を持つ。初期化時、Ｄ１１バッファーは通常モード（非トグル状態）であるが、この関数を呼び出す度トグル状態、非トグル状態を繰り返す。

【20】アナログ入力モードの設定

int __stdcall AD686z_Set_Inpmode(int inp_mode, int ad_code, int ad_reso, int ad_rangemode);	
引数	inp_mode:アナログ入力信号形式（ダミー） ad_code:ＡＤデータコード指定　０：バイナリ、１：２の補数 ad_reso:ＡＤ変換分解能（ダミー） ad_rangemode:ダミー
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	アナログ入力変換コードを指定する

【21】サンプリング動作の強制停止

int __stdcall AD686z_Stop_Samp(void);	
引数	なし
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	サンプリング動作を強制的に停止する。内部メモリに存在した残りデータも廃棄される。

【22】各フラグのクリア

int __stdcall AD686z_Clear_Flags(int c_data);	
引数	c_data:クリアビット指定データ（３－１２項参照）
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	ＡＤボード上の各フラグをクリアする

【23】残りＡＤデータの読込み

int __stdcall AD686z_Read_RestData(void);	
引数	なし
戻り値	正常終了時：０（残りデータが無かった場合）、 正の整数（残りデータ数：総チャンネル分） エラー発生時：エラーコード（負の値）
機能・動作	Ｄ１１バッファー使用サンプリングの場合にデータロストエラーが発生した場合、ボード内に残存データがあれば、当関数を使用してＤ１１バッファーへ追加する事ができる。

【24】外付・同時サンプルホールド制御

int __stdcall AD686z_Set_Shc(int board_no, int shc);	
引数	board_no:対象ボード番号（ボード毎に設定を行う） shc:外付・同時サンプルホールド制御　０：しない、１：する
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	同時サンプリング回路のためにホールド時間（1uS）をサンプリングスキャン先頭に追加するかどうかを制御する。shc を“０”とするとサンプリング速度低下はないが、その代りスキャン先頭チャンネルについて数 mV のホールドステップ誤差が発生する。

【25】サンプリングスキャン速度指定

int __stdcall AD686z_Set_ScanSpeed(int scan_speed);	
引数	scan_speed:サンプリングスキャン速度指定　０：低速（10uS/Ch）、１：高速（5uS/Ch）
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	サンプリングスキャン速度を指定する

【26】オプションメッセージ発信の設定

int __stdcall AD686z_Plus_Message(int submit);	
引数	submit:オプションメッセージ発信の有無指定　０：発信しない、１：発信する
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	マニュアルモードの場合、ＡＤデータ読込みタイミングフラグ（trs_trig で指定されるフラグ）を検出した時点でメッセージを送信する。 自動モードの場合、ＡＤデータ読込みタイミングフラグ（trs_trig で指定されるフラグ）によりデータ転送が終了した段階でメッセージを発信する。

【27】本ハンドラーのバージョン取得

int __stdcall AD686z_Get_Libver(int ver);	
引数	ver:　０：戻り値はメジャーバージョン+マイナーバージョン １：戻り値はメジャーバージョン ２：戻り値はマイナーバージョン
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	本ハンドラーのバージョン番号を得る

FCAD516 独自機能用関数セット

以下の関数は FCAD516PCI に導入されたハードウェアを操作する専用関数群である。

【28】ハーフフルサイズの取得

int __stdcall AD516_Get_HalfSize (void);	
引数	なし
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	ハーフフルサイズを取得する関数

【29】使用済みメモリ量の取得

int __stdcall AD516_Get_UsedSize (UINT_PTR *mem_size);	
引数	mem_size:メモリ使用量を（ワード単位で）取得するためのポインター
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	使用済みメモリ量をポインターに得る関数

【30】アナログ回路条件の設定

int __stdcall AD516_Set_DataCode (int board_no, int *se_diff, int *sence_level, int *range_mode);	
引数	board_no:操作対象ボードを規定する SW-BN スイッチの値 se_diff:入力接続切り替え（０：差動入力、１：シングルエンド入力） sence_level:入力感度切り替え（０：０～＋５Ｖ、１：０～＋１０Ｖ、２：±２．５Ｖ、 ３：±５Ｖ、４：±１０Ｖ） range_mode:A レンジ・B レンジ切り替え（０：A レンジ、１：B レンジ） 注）board_no を除きポインター渡し
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	Board_no で指定するボードについて入力モード、データコード等を指定する関数

【31】AD データコード、入力モード等の取得

int __stdcall AD516_Get_DataCode(int board_no, int *se_diff, int *sence_level,int *range_mode);	
引数	board_no:操作対象ボードを規定する SW-BN スイッチの値 se_diff:入力接続切り替え（０：差動入力、１：シングルエンド入力） sence_level:入力感度切り替え（０：０～５Ｖ、１：０～１０Ｖ、２：±２．５Ｖ、 ３：±５Ｖ、４：±１０Ｖ） range_mode:A レンジ・B レンジ切り替え（０：A レンジ、１：B レンジ） 注）board_no を除きポインター渡し
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	Board_no で指定するボードについて入力モード、データコード等を取得する関数

４－７　エラーコード

戻り値	不具合の内容	適用関数等
－１	必要な PCI ライブラリが見つからない	AD686z_Open_ADSys
－２	FCAD516PCI ボードが見つからなかった	各関数
－３	マスターボードが見つからなかった	各関数
－４	SW-BN スイッチが重複している	AD686z_Open_ADSys
－５	データコード指定が間違っている	AD686z_Set_Inpmode
－６	指定されたユニット番号のボードが存在しなかった	各関数
－７	指定されたボード数と実際の枚数が異なっていた	AD686z_Open_ADSys
－８	オープンされているボードがない	各関数
－９	スタートしているボードがない	各関数
－１０	指定チャンネル数が異常	AD686z_Set_SampCh
－１１	転送フラグ指定エラー	AD686z_Set_SampMode
－１２	バッファー指定エラー	AD686z_Set_SampMode
－１３	トリガ選択エラー	AD686z_Set_Trigger 等
－１４	トリガ源選択エラー	AD686z_Set_Trigger
－１５	トリガのレベル関係が異常	AD686z_Set_RangeTrigger
－１６	外部クロック値設定エラー	AD686z_Set_Exclk 等
－１７	クロックソース指定が異常	AD686z_Set_Clock,AD686z_Set_MsClock
－１８	クロック周期指定が異常	AD686z_Set_Clock,AD686z_Set_MsClock
－１９	スキャン周期指定が仕様範囲外	AD686z_Set_Clock
－２０	分周比指定が許容範囲外	AD686z_Set_Clock
－２１	クロック指定方法が間違っている	AD686z_Set_Clock,AD686z_Set_MsClock
－２２	Dll バッファポインターがおかしい	AD686z_Read_DllData
－２３	Dll バッファサイズがおかしい	AD686z_Read_DllData, _Get_OneScan
－２４	何らかの理由で動作していないボードがある	AD686z_Get_OneScan
－２５	**スキャン速度指定エラー**	AD686z_Set_ScanSpeed
－２６	バージョン番号指定エラー	AD686z_Get_Libver
－２７	**メッセージ送信変数エラー**	AD686z_Plus_Message
－２８	アナログ設定設定値エラー	AD516_Set_DataCode
－２９	チャンネル数がボード毎に異なっていた	AD686z_Start_Samp
－３０	既にサンプリング中	各関数
－３１	サンプリング数が異常だった	AD686z_Start_Samp
－３２	ユーザーバッファーが指定されている	AD686z_Read_RestData
－３３	複数ボード間でハーフフルサイズが異なっていた	AD686z_Open_ADSys
－３４	データロストが発生していない	AD686z_Read_RestData
－３５	メモリ確保に失敗した	ブロックリード命令関連
－３６	外部クロックが設定されていない	AD686z_Set_Clock
－３７	入力接続がボードによって異なっている	AD516_Set_DataCode
－３８	サンプルホールドパラメータ異常	AD686z_Set_Shc
－３９	平均化処理実行不可（平均化回数過多）	AD686z_Set_MultiScan

第５章　汎用デバイスドライバー仕様書

本仕様書は、フレックスコア製 PCI ボードを制御するために必要となる汎用デバイスドライバーの内容について記述したものです。本デバイスドライバーは、最大で 16 台までの PCI ボードを同時に制御する事が可能ですが、基本的な仕組みは、ドライバーオープン及びドライバースタートから始まり、夫々の PCI ボードリソースを、Ｉ／O制御レジスタへのアクセスにより制御し、全ての制御が終了した段階でドライバーストップ、更にドライバークローズ処理を行って制御全てを終了するところまでを行います。
基本的なリソース制御については、バイト制御、バイトブロック制御、ワード制御、ワードブロック制御、更にダブルワード制御及びダブルワードブロック制御の１２種類（各々リード・ライト処理がある）が定義されています。実際には、本仕様書脱稿の段階でダブルワード制御を行うレジスタを装備したボードは存在しませんが、将来的にそのようなボードが成立する可能性も充分考えられるため、ハンドラー仕様書として内容を整備しています。

本ドライバーは、Windows7 等での PCI ボードインストールとの関係上、対応する専用ドライバーインストール時に共にインストールされるよう設定されています。（即ち専用のインストーラーは存在しません。）

また、本ドライバーは、上記の如く単純なＩ／Ｏレジスタアクセスのみから構成されている為、基本的に無サポートとします。また全てのデータ処理（バイト／ワード／ダブルワード：リード／ライト）については、クリティカルセクションによって排他処理が保証されています。

※　クリティカルセクションについては、特にマルチコアＣＰＵ等の環境で本ドライバーを使用する際に必要となるものですが、ここではその詳細については記述しません。詳細については、マイクロソフト製ビジュアルスタジオ等に詳細な記述があるのでこれを参照してください。

ハンドラー及び関連ファイル名称一覧表

ＯＳ及びビット幅	Windows7　32 ビット	Windows7　64 ビット	WindowsXP　32 ビット
ドライバー名称	PORT7_X86.Dll	PORT7_X64.Dll	PORT7_X86.Dll
インポートライブラリ名称	PORT7_X86.Lib	PORT7_X64.Lib	PORT7_X86.Lib
関数インデックス	PORT7.H	PORT7.H	PORT7.H

上記ドライバー３種類の内、WindowsXP32 ビット用ドライバーについては、システム移行動作確認用としてのみ使用する事を前提として公開しています。このドライバーは、WindowsXP システムで動作していたアプリを Windows7 へ移行する際に、その操作を段階的に（つまり OS の変更を行うこと無しに）新しいドライバーでの動作評価を行うことができるよう用意されているものです。（ハンドラーの内容としては Windows７　３２ビット版と全く同じ構成になっています。）

以下に記述する仕様は、ある固有ベンダーID に従った PCI ボード群に対する汎用デバイスドライバーの仕様書です。ベンダーID は固有の値（現状０ｘ１３ＦＤ）でなければなりません。

【1】ドライバーオープン

int __stdcall Open_Driver(int *device_id);	
引数	device_id:各ボードのデバイス ID をセットする配列（ユーザー側にて確保:最大 16 個）
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	ハンドラー内部の初期化を行い、システム内で検出されたそれぞれのボードのデバイス ID を報告する。

【2】ドライバースタート

int __stdcall Start_Driver(int board_no, int base_no, int *resorce_no);	
引数	board_no: ドライバーオープンで報告された、デバイス ID を格納している配列の要素番号であり、尚且つシステム内で検索され発見された該当ボードの発見順序を表す変数である base_no: ＩＯリソース番号（２固定） resource_no:当汎用ハンドラー内で確定したリソースを指し示す要素番号
戻り値	正常終了時：０ エラー：エラーコード（負の値）
機能・動作	ドライバーオープンで検出されたボード群の内 board_no で指定したボードと通信を開始する。

【3】バイトリード

int __stdcall ReadByte(int resource_no, int offset);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス
戻り値	正常終了時：読み込んだバイト値 エラー発生時：エラーコード（負の値）
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタの値を読み込みユーザーへ返す関数（戻り値＝バイト）

【4】バイトブロックリード

int __stdcall ReadByteBlock(int resource_no, int offset, int count, BYTE *pbuf);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス count:読込みを行うバイト数 pbuf:読み込んだレジスタからの値を保存する先頭アドレス（ポインター）
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタの値を count 回数読み込みユーザーが用意したデータバッファ（pbuf）へ返す関数である。pbuf データバッファのサイズは count（バイト）必要である。

【5】ワードリード

int __stdcall ReadWord(int resource_no, int offset);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス
戻り値	正常終了時：読み込んだワード値 エラー発生時：エラーコード（負の値）
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタの値を読み込みユーザーへ返す関数

【6】ワードブロックリード

int __stdcall ReadWordBlock(int resource_no, int offset, int count, WORD *pbuf);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス count:読込みを行うワード数 pbuf:読み込んだレジスタからの値を保存する先頭アドレス（ポインター）
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	board_no で示されるボードの offset アドレスに存在するレジスタの値を count 回数読み込みユーザーが用意したデータバッファ（ret_value）へ返す関数である。Ret_value データバッファのサイズは count×２（バイト）必要である。

【7】ダブルワードリード

int __stdcall ReadDWord(int resource_no, int offset);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス
戻り値	正常終了時：読み込んだダブルワード値 エラー発生時：エラーコード（負の値）
機能・動作	board_no で示されるボードの offset アドレスに存在するレジスタの値を読み込みユーザーへ返す関数

【8】ダブルワードブロックリード

int __stdcall ReadDWordBlock(int resource_no, int offset, int count, UINT32 *pbuf);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス count:読込みを行うワード数 pbuf:読み込んだレジスタからの値を保存する先頭アドレス（ポインター）
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタの値を count 回数読み込みユーザーが用意したデータバッファ（pbuf）へ返す関数である。pbuf データバッファのサイズは count×４必要である。

【9】バイトライト

int __stdcall WriteByte(int resource_no, int offset, BYTE data);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス data:レジスタへ書き込む値
戻り値	正常終了時：0 エラー発生時：エラーコード（負の値）
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタへ data を書き込む関数

【10】バイトブロックライト

int __stdcall WriteByteBlock(int resource_no, int offset, int count, BYTE *pbuf);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス count:読込みを行うワード数 pbuf:読み込んだレジスタからの値を保存する先頭アドレス（ポインター）
戻り値	正常終了時：0 エラー発生時：エラーコード（負の値）
機能・動作	board_no で示されるボードの offset アドレスに存在するレジスタへ pbuf バッファーから count 回数書き込みを行う関数である。Pbuf データバッファのサイズは count（バイト）必要である。

【11】ワードライト

int __stdcall WriteWord(int resource_no, int offset, WORD data);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス data:レジスタへ書き込む値
戻り値	正常終了時：0 エラー発生時：エラーコード（負の値）
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタへ data に保存されているデータを書き込む関数

【12】ワードブロックライト

int __stdcall WriteWordBlock(int resource_no, int offset, int count, WORD *pbuf);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス count:書き込みデータ数 pbuf:レジスタへ書き込む値を保存しているアドレス（ポインター）
戻り値	正常終了時：0 エラー発生時：エラーコード（負の値）
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタへ pbuf に保存されているデータを書き込む関数。pbuf データバッファのサイズは count×２（バイト）必要である。

【13】ダブルワードライト

int __stdcall WriteDWord(int resource_no, int offset, UINT32 data);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス data:レジスタへ書き込む値
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタへ data に保存されているデータを書き込む関数

【14】ダブルワードブロックライト

int __stdcall WriteDWordBlock(int resource_no, int offset, int count, UINT32 *pbuf);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス count:書き込みデータ数 pbuf:レジスタへ書き込む値を保存しているアドレス（ポインター）
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタへ pbuf に保存されているデータを書き込む関数。pbuf データバッファのサイズは count×４（バイト）必要である。

【15】ドライバーストップ

int __stdcall Stop_Driver(int resource_no);	
引数	resource_no:　ドライバースタート関数から返されたボードリソースインデックス
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	全てのユニットで使用していたメモリリソースを開放し通信を終了する。

【16】ドライバークローズ

int __stdcall Close_Driver(void);	
引数	なし
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	全てのボードについてハンドルを開放する。

エラーコード

戻り値	内容
－１	必要な PCI ライブラリが見つからない
－２	ライブラリのデバッグオプションが無効だった
－３	PCI ボードが見つからなかった
－４	ボード番号指定がシステム上の枚数を上回っている
－５	オープンしていないボードをクローズしようとした
－６	定義されていないベースナンバーを参照しようとした
－７	オープンしていないボードに対してアクセスが行われた
－８	リソース範囲不適
－９	内部メモリ確保エラー

第６章　保守、その他

６－１　故障・トラブル等の原因と対処

本ボードは、ＤＯＳ／Ｖパソコンで全数検査の上出荷されています。
お手元での動作確認方法は１－６項に記されています。動作に不具合がある時は、以下の諸点を再点検して下さい。
それでも不明なときは、システム構成等をご記入の上、弊社品質管理部まで御連絡下さい。情報伝達手段としてfax及びEmailを準備しておりますが、ご使用しやすい方の手段をお選びの上、御連絡願います。
一般的に言って、応答速度を求められるときはEmailが、複雑な情報を含む場合はfaxがお勧めだと思います。

再点検、確認ポイント

ボードID

単体で動作させる時、この番号が０以外の設定になっているとハンドラーがユニットを認識しません。実際に起こり得る局面としては、マスタースレーブ接続で使用されていたスレーブユニットを、単独で使用する等の局面が考えられます。

トリガ方法

アナログレベルトリガは、アナログ変換値の値で、アナログエッジトリガは、アナログ変換値の値の変化で、夫々トリガを認識します。そのため、アナログレベルトリガでは、サンプリング開始と同時にトリガ検出が行われることがある一方で、アナログエッジトリガでは、アナログ電圧値がスレッショルドをよぎる変化を起こさないとトリガとして認識しません。詳細については、３－６項を参照して下さい。

デジタル入出力

本ボードのデジタル入力に接続できる信号源はTTL(LS,CMOS等の５Ｖ電源動作素子)に限ります。
それ以外の電源電圧を用いている信号源を接続すると、殆ど一瞬のうちに入力素子が破壊します。
出力についてもTTL信号源以外の信号源に接続することは、即出力素子の破壊を招きます。
本ボードで使用しているデジタル入出力素子の絶対最大定格は－０．６Ｖ～＋７．０Ｖとなっています。

アナログ入力

本ボードの過電圧入力保護は±３５Vが最大です。そのため一瞬であってもこの電圧範囲を超える電圧が印加されるとアナログ入力回路素子の破壊につながります。また、アナログ回路は、デジタル回路と異なり異常電圧が伝播していく可能性が高いため、異常電圧による被害はデジタル回路に比べて大きくなりがちです。

複数チャンネル使用時の各信号源のグランド電位差にも注意が必要です。特に様々な機器からの入力を接続する機会が多いこともあいまって、この原因による回路破壊も時折見られます。予防策としては、夫々の回路を接続する前に、機器間のグランド電位差をテスターなどで確認しておく事があげられます。一般的には交流的な電位差がある場合が多いと考えられますが、直流的な電位差についても確認が必要です。

更に、本ボードのグランド電位は登載されているパソコンのグランド電位と共通であるという点にも注意が必要です。

６－２　修理のときは

本製品使用上の質問・トラブル対応・故障修理などは入手経路の如何にかかわらず、弊社宛に直接御相談下さい。
商社等を経由されても弊社として問題はありませんが、情報交換の密度、修理完了までの時間などの面で多少不利になるのではないかと思われます。
また、導入当初からの不具合については、誤解や情報不足による事が多いので、事前にご相談下さい。
無償修理と有償修理の場合分けにつきましては以下の通りとなります。
納入後１年間は自然故障、及び弊社製造上の問題に起因した事が明らかな故障製品に対して無償修理を行います。

但し保証期間中であっても、次の場合は有償修理となります。

ア　取扱上の不注意、誤用による故障および損傷
イ　弊社以外での修理、改造、分解掃除等による故障および損傷
ウ　泥、砂、水などのかぶり、落下、衝撃等が原因で発生した故障および損傷
エ　火災、地震、水害、落雷その他の天変地異、公害や異常電圧による故障および損傷
オ　保管上の不備（高温多湿の場所等）や手入れの不備による故障
カ　接続している他の機器に起因して故障が生じた場合
キ　その他使用者側の責に帰する故障

また、不良要因が再現されなかった場合につきましては、着払いにより製品を返却させて頂く場合もありますのでご了承下さい。

修理は宅配便によるセンドバックで行います。尚運賃は互いに発送する側が負担するものとします。
出張修理は行っておりません。簡単な故障であれば一週間程度で修理可能ですが、故障状況によっては更に日数を要する事もあります。
また、有償修理の場合の修理料金については、簡単な故障で基本料金￥１０，０００円＋部品代とお考え下さい（事務手数料を含む）。修理費用限度額がある場合は、お申し出頂ければ超過する事が明確になった時点で御連絡いたします。

修理品送付先
〒301-0853
茨城県龍ヶ崎市松ヶ丘３丁目１８番地３
フレックスコア
品質管理部　林
E-mail: support@flexcore.jp
Fax:050-3488-3354

６－３　アナログ入力範囲の再調整

１－６項に示すような形で動作確認を行い、入力範囲やオフセットの変化が認められるときは再調整が必要です。 アナログ回路は経年環境変化に対する保守を定期的に行うことが望ましく、夏冬の使用環境・周囲温度に差がある場合は季節単位、通年安定した使用環境の場合は１～２年に１度は校正することが理想的です。

再調整の方法・手順を以下に記しますが、被調整対象のＦＣＡＤ５１６ＰＣＩ－１６MWの他に、高確度の基準電源と高精度の電圧計、各入力を個別に切り替える事ができるスイッチボックス及びWindow XP／Vistaがインストールされたパソコンを準備する必要があります。

６－４　外部制御信号、タイミング等